Optimization of Support Vector Machines for balancing the trade-off between generalization performance and computational complexity

Submitted in partial fulfillment of the requirements

for the award of the degree of

DOCTOR OF PHILOSOPHY

Submitted by

Lavanya Madhuri Bollipo

(Roll No. 701433)

Under the supervision of

Dr. Kadambari K. V.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL
TELANGANA - 506004, INDIA
August 2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL
TELANGANA - 506004, INDIA

THESIS APPROVAL FOR Ph.D.

This is to certify that the thesis entitled, **Optimization of Support vector machines for balancing the trade-off between generalization performance and computational complexity**, submitted by **Ms. Lavanya Madhuri Bollipo [Roll No. 701433]** is approved for the degree of **DOCTOR OF PHILOSOPHY** at National Institute of Technology Warangal.

Examiners

Research Supervisor	**Chairman**
Dr. Kadambari K. V.	**Prof. P. Radha Krishna**
Dept. of Computer Science and Engg.	Dept. of Computer Science and Engg.
NIT Warangal	NIT Warangal
India	India

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL
TELANGANA - 506004, INDIA

CERTIFICATE

This is to certify that the thesis entitled, Optimization of Support Vector Machines for balancing the trade-off between generalization performance and computational complexity, submitted in partial fulfillment of requirement for the award of degree of DOCTOR OF PHILOSOPHY to National Institute of Technology Warangal, is a bonafide research work done by Ms. Lavanya Madhuri Bollipo [Roll No. 701433] under my supervision. The contents of the thesis have not been submitted elsewhere for the award of any degree.

Research Supervisor
Dr. Kadambari K. V.
Dept. of Computer Science and Engg.
NIT Warangal
India

Place: Warangal

Date: 03-08-2021

DECLARATION

This is to certify that the work presented in the thesis entitled "*Optimization of Support Vector Machines for balancing the trade-off between generalization performance and computational complexity*" is a bonafide work done by me under the supervision of Dr. Kadambari. K V. The work was not submitted elsewhere for the award of any degree.

I declare that this written submission represents my ideas in my own words and where others ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/date/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Lavanya Madhuri Bollipo

(Roll No. 701433)

Date: 03-08-2021

ACKNOWLEDGMENTS

First and foremost, praises and thanks to the GOD, the Almighty, for HIS showers of blessings throughout my research work to complete it successfully.

It is an immense pleasure to express my sincere thanks and deep sense of gratitude to my supervisor Dr. Kadambari K V, Assistant Professor, Department of Computer Science and Engineering, National Institute of Technology (NIT) - Warangal for giving me the opportunity to do research under her and providing valuable guidance throughout this research. Her dynamism, vision, sincerity, love and motivation have deeply inspired me to work hard to carry out this research successfully. It is a great privilege and honor to me to have her as my research supervisor. I extend my gratitude to the Doctoral Scrutiny Committee (DSC) members comprising of Prof. P. Radha Krishna, Prof. S. G. Sanjeevi, Prof. D. Srinivasacharya and Dr. Ch. Sudhakar for their insightful comments and suggestions during oral presentations. I am extending my sincere gratitude to (late)Dr. Jagannath Sardar, Scientist-C (R&S), TTL, Varanasi,CSTRI, Central Silk Board, Govt. of India, IIHT Campus, Chowkaghat, Varanasi-221002 for his valuable suggestions which helped in paper publications. I am also thankful to the office and technical staff for helping me out throughout my PhD in providing valuable support and technical help.

I am extremely grateful to my parents Mr. B. Nageswara Rao and Mrs. Santhosham for their love, prayers, caring and sacrifices throughout my life. I am very much thankful to my husband, Mr. K. Sandeep for his sacrifice and constant encouragement throughout this research. May GOD bless my chidlren Anshi and Teddy for being so understanding while I was doing my PhD work. Also, I express my thanks to my sisters Ms. Jyothi and Ms. Bindu for their support and prayers.

Lavanya Madhuri Bollipo

Dedicated to

My Family

ABSTRACT

The ever growing volumes of data, availability, and its complexity demands for development and adoption of better data analytic technologies to uncover hidden patterns and correlations in it. The better way to handle this process is through machine learning (ML). ML is a collection of data analysis techniques based on artificial intelligence (AI) that allows to design more accurate algorithms in predicting new data trends from historical data. The most widely used ML paradigms are supervised and unsupervised learning strategies. Among the various existing ML algorithms, Support vector machine (SVM) for prediction problems gained profound interest due to its special abilities. SVM always tries to achieve unique global minima and also scales linearly to high dimensional data. There is even less risk of overfitting and has good generalization performance in practice. Yet, the scalability and performance of nonlinear SVM classifiers on continuous heterogeneous data is challenging. Nonlinear SVM mitigates with issues like data piling, outliers, overfitting and imbalanced data. Besides this, computational complexity of nonlinear SVM can reach upto $\mathcal{O}(n^3)$ when the number of iterations scales up with the number of training samples.

To reduce the computational overheads, batch learning of SVM can be replaced with incremental learning that can lead to cheaper iteration cost. However, the drawback of incremental learning is its slow convergence rate and long training time. To enhance the generalization performance, several iterative optimization techniques are proposed. The optimization techniques chosen can improve the generalization ability and also reduce the computational complexity.

The thesis proposes to optimize some of the variants of SVM to balance the trade-off between generalization performance and computational complexity when applied to incremental data used in classification and regression applications. The first work of the thesis focuses on optimizing SVM using modified Frank-Wolfe algorithm (SVM-MFW). The proposed model is applied on classification and prediction of Parkinson's disease in its early stages. SVM-MFW makes use of an away-step technique which in a way only increases the weights of the vectors corresponding to the optimal solution and also discards the spurious points. The proposed method converges to the optimal solution in less number of iterations thereby producing sparser representation of support vectors. Consequently, SVM-MFW algorithm learns incrementally by providing computationally simpler updates to train, and accelerates the convergence rate as well. The accuracy achieved by the algorithm is 98.3%, and prediction accuracy is evaluated using cross-entropy, which is 0.134 and CPU time: 2.32 sec.

In the second work, ε-Support vector regression algorithm is optimized with Large margin distribution (LDM) technique to enhance the performance of the ε-SVR (ε-MDSVR). To evaluate the model, experiments are performed on four benchmark datasets taken from UCI repository. The proposed model reduces the scattering of data in ε-tube by utilizing the whole training dataset to avoid over-fitting. The proposed method achieves a better generalization performance by optimizing the margin distribution which is done by employing modified Dual coordinate descent method (DCD). The modified DCD selects one variable in the solution space to update at each iteration which possibly generates the maximum optimization in the objective function. The learning speed and generalization performance can be improved by integrating LDM and DCD techniques. Experimental results are validated on popular matrices of Mean square error (MSE) and Correlation coefficient (R^2). It is observed that the proposed algorithm significantly achieves good predictive accuracy with low error rate and high correlations when compared to the classical SVR and other regression techniques.

The third work optimizes ϑ - Support vector regression using bounded functions on noisy datasets (ϑ-PSVR). The proposed work brings a balance between the number of support vectors and errors which is trained over Parkinson's dataset obtained from PPMI. The algorithm addresses the key issues of data piling and overfitting effect on noisy data which effects the generalization performance and computational complexity in case of classical ϑ-SVR. The proposed method takes all data into consideration having direct impact on weight vector w and uses two bounded functions on the data which is perturbed by noise. The weight vector w is used to find the position of training points to the fitting curve with respect to its functional margin. This algorithm tries to generate a more smoother regression curve and achieves better prediction accuracy with MSE=0.131 and R^2=0.758 in less computation time of 2.26 sec compared to classical ϑ-SVR.

The fourth work details the acceleration of incremental learning and decremental unlearning of support vector machines used on imbalanced datasets. The proposed model combines weak SVM classifiers with asymmetric misclassification cost to modify the training datasets. This modified dataset is used to boost the prediction accuracy of weak SVMs at each iteration during learning. Later, the outcomes of these boosted SVMs are integrated by a weighted majority vote to generate final class label. The proposed algorithm increases the learning rate and raises the predictive accuracy of incremental and decremental SVM.

Keywords: Support vector machines, Incremental Learning, Computational Complexity, Frank-Wolfe algorithm, Noisy data, Large margin distribution, Coordinate descent, Boosting.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This Chapter introduces the popular support vector machines among data analysis techniques and discusses some of SVM variants. The chapter also provides motivations behind the work documented in the thesis. The aims and contributions of the thesis are outlined, and also, the contents of each chapter are briefly described.

1.1 Machine Learning

The availability of large volumes of heterogeneous data, cheaper and powerful computational processing, and affordable data storage nowadays demands to develop automatic data analysis models that can handle complex data and produce accurate results quickly [1]. Machine learning (ML) is a collection data analysis tools based on Artificial Intelligence (AI). ML is developed to make computers automatically learn from data to generate its structure and relationships among it even if the information of data trends are not available [2, 3]. Working of ML algorithms involves training over a input data to create a model, test it with a new sample to make a prediction, and evaluate its performance. If the performance is not anticipated, the model trains continuously until the expected outcome is obtained. This process makes the ML algorithm to train automatically until an optimal prediction is generated that will fairly improve accuracy of the model overtime. ML algorithms often use an iterative approach to implement its training process where passes are run through the data until a vigorous pattern is found. The test for a ML model is a validation error on new data from historical data [4]. After a training phase, ML can derive associative meaning to the data, and therefore distinguish samples from another as shown in Figure 1.1. Therefore, ML techniques provides smart alternatives to analyze vast data and gives the system the ability to learn from data without being explicitly programmed. By developing fast and efficient algorithms and data-driven models for real-time data processing, it can produce accurate results and analysis at faster rates [5]. ML utilizes a variety of techniques to handle such complex data to make predictions better [3, 6]. Nowadays, machine learning powers many applications in every side of our lives [7].

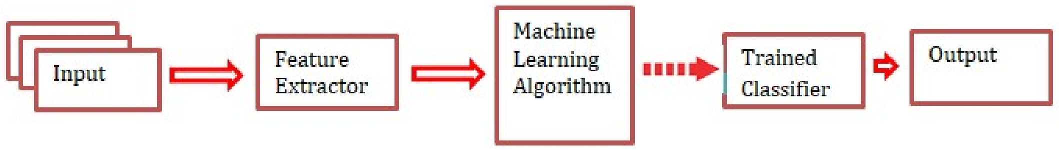


Figure 1.1: Machine Learning Phases

The most extensively popular machine learning methods is supervised and unsupervised learning [8]. Supervised learning models are mostly used in practice [9]. Supervised learning attempts to train from given historical data and its relationship among themselves. unsupervised learning attempts to produce the patterns from the given data without knowing its structure. Mathematically, in supervised learning, input and output/target variables (x, y) are given and algorithm has to derive the mapping function $y = f(x)$ from the input to the output. Supervised learning can be further divided into classification and regression problems [10, 11]. Classification treats the output variable as a category label, where as regression treats the output variable as a real value. In this study, we limit our focus to the popular supervised machine learning algorithm called as Support vector machines (SVM) [12]. SVM always tries to achieve unique global minima and also scales linearly to high dimensional data. There is even less risk of over-fitting and has good generalization performance in practice. Over the past several years, SVM is used extensively for the analysis of classification and prediction problems of many practical applications [13, 14].

1.2 A breif about SVM

The SVM was developed by V. Vapnik in the 90's [12]. It is an optimization algorithm based on the separating hyperplane. SVM is originally designed for classification of data. This classification process consists of two phases: Learning: Training SVM algorithm with data samples at its disposition. Classification: New data samples for which the result is not known is used to test the algorithm's performance. SVM analyzes the data to produce results which are more correlated to those trained data samples used in the learning phase. Each data sample is represented as a pair of (input, output), where input is the given dataset and output is generally denoted as a class label which tells the trends and relationship of data. If the data has high dimensionality, the different classes, which constitute different clusters, are linearly separable by the hyperplanes. The kernel trick permits to deal with linearly inseparable surfaces. Kernel trick transposes non-linear data into higher dimensional space (called the feature space) to find the linearly separating hyperplane. In simple words, SVM is an algorithm that takes the data and its labels as an input and outputs a line that can separate data based on corresponding labels. SVM's key idea is to produce a best decision surface that can distinguish data as far as possible. The Figure 1.2 shows the datasets with two class labels Red and Blue, SVM finds the data samples which are nearer to the separating line from both the classes (also called support vectors). SVM measures the distance between the separating line and the support vectors. This distance is known as the margin. SVM's goal is to maximize the minimum margin of data vectors and produces a optimal hyperplane with highest margin between these data points. If the data is linearly separable, a linear SVM is implemented, if data is not linearly separable then Non-Linear SVM is used.

Let T be the training data with n sample points. Each data point is denoted by $\{(x_i, y_i)\}$, $i = 1$ to n, where $x_i \in \mathbb{R}^m$ is the input vector of $n \times m$ dimensions and $y_i \in \mathbb{R}$ is the corresponding output vector of $n \times 1$ dimensions. SVM generates a decision surface which distinguishes the input space with respect to its target values. Assume that training samples are (x_i, y_i) where $x_i \in \mathbb{R}^m$ and for each sample, a label $y_i \in (+1, -1)$ represents

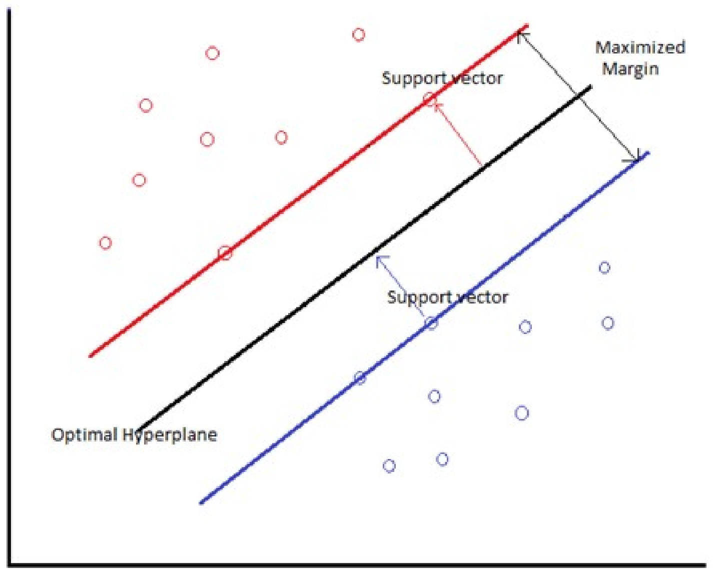


Figure 1.2: SVM Optimal Margin

to which of the two classes the sample belongs. Consider the Figure 1.2, where SVM tries to separate two classes of data i.e., Red and Blue. Learning SVM can be formulated as an optimization, which tries to obtain maximum separation among the classes of data. The distance between the upper margin (Red line) to lower margin (Blue line) is the length of the ε-tube. The black line is the optimal hyperplane which distinguishes two data classes. SVM is an optimization problem which outputs optimal hyperplane equation. The line equations of both upper and lower margin can be stated as: $w_i.x_i + b = -1, \quad w_i.x_i + b = +1$. w_i is a weight vector, $\frac{2}{\|\mathrm{w}\|}$ is the separation among two classes, called as the width of the and decision curve and SVM seeks to increase this witdh, which is same as minimising its norm:

$$\max_{w} \quad \frac{2}{\|\mathrm{w}\|} \qquad \text{subject to} \qquad \mathrm{wx}_i + b \geq +1 \quad \text{if} \quad y_i = +1; \forall i = 1 \ldots n.$$
$$\mathrm{wx}_i + b \leq -1 \quad \text{if} \quad y_i = -1; \forall i = 1 \ldots n.$$

These two constraints can be represented in one inequality: $\mathrm{y_i}\left(w^T x_i + b\right) \geq 1$

SVM seeks to maximize this objective function, which is equivalent to minimising its norm:

$$\min_{w} \quad \|\mathrm{w}\|^2 \quad s.t. \quad y_i\,(\mathrm{wx}_i + b) \geq 1 \quad \forall i = 1 \ldots n. \tag{1.1}$$

The Equation (1.1) is a primal form of quadratic optimization problem subject to linear parameters and there is a unique global optimum value. The classification function is:

$$\mathrm{y}_{\text{new}} = \operatorname{sign}(\mathrm{w}x_{\text{new}} + b) \tag{1.2}$$

In primal form, there will be high computation burden due to high-dimensionality and many parameters are to be solved. These constraints can be easily solved in dual form of SVM. Dual formulations assigns parameters to samples only, but not to features. Apply Lagrangian function to SVM according to [15],we get:

$$L(\mathbf{w}, \alpha) = \frac{1}{2}\|\mathrm{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left[(\mathbf{w} \cdot \mathbf{x}_i + b)\, y_i - 1\right] \quad s.t. \quad \alpha_i \geq 0, \forall i \tag{1.3}$$

To find the w and b values, take the derivative of each constraint.

$\frac{\partial L}{\partial w} = w - \sum_i^n \alpha_i y_i x_i$

$\mathbf{w} = \sum_i^n \alpha_i y_i \mathbf{x}_j; \quad \alpha_i \geq 0.$

$\frac{\partial L}{\partial b} = -\sum_i^n \alpha_i y_i$

$\sum_i^n \alpha_i y_i = 0$

Substituting these values back in Equation (1.1) (and simplifying), we obtain:

$$\max_{\alpha} \quad \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i . x_j \tag{1.4}$$

where, $\sum_i \alpha_i y_i = 0 \quad \alpha_i \geq 0$

The classification equation is:

$$y_{\text{new}} = \operatorname{sign}\left(\sum_{\mathrm{i}} \alpha_i \mathrm{y}_{\mathrm{i}} \mathbf{x}_{\mathrm{i}} \mathbf{x}_{\text{new}} + b\right) \tag{1.5}$$

1.2.1 Soft-margin SVM

The soft-margin SVM allows some misclassification of data points deviated from decision curve to certain region. This region is called ε-insensitivity zone and the allowed deviation is called ε-insensitivity tube. Soft-margin SVM introduces a error cost parameter ξ_i in the objective function $f(x)$ to measure the classification error for the data points that lie within the ε-insensitivity tube. Therefore, instances that lie within certain margin (denoted by ε) around the curve do not incur any cost. Intuitively, this means that some errors are also allowed as long as they are within an ε-deviation of $f(x)$ and is known as ε-insensitive loss function. SVM tries to find a function $f(x)$ to fit all training samples such that its divergence is not exceeding ε. The use of kernel mapping on input space allows $f(x)$ to dependent on number of support vectors instead of its dimensions.

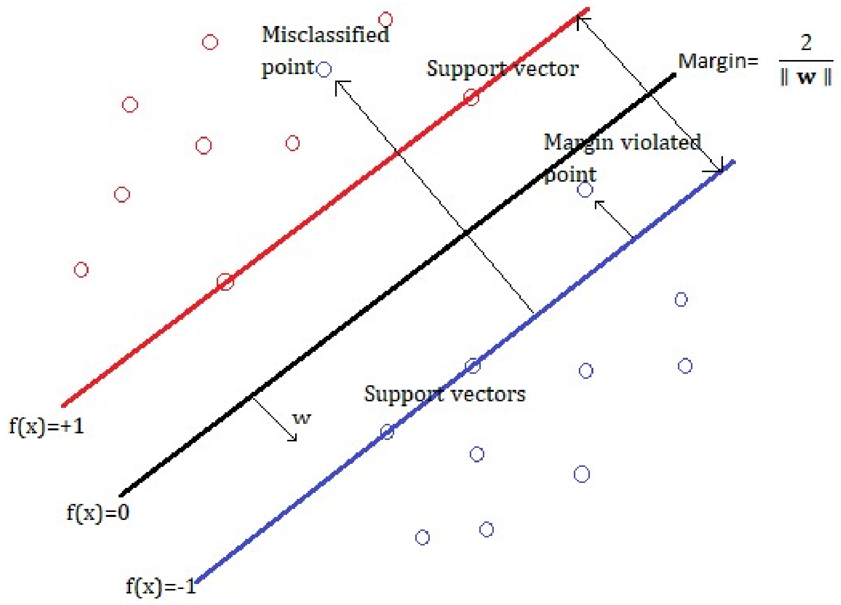


Figure 1.3: Soft margin SVM

Consider the Figure 1.3, to get soft margin, use the slack variable, (ξ) to approximate the number of misclassified points.

1. For $0 < \xi \leq 1$ point lies within ε-tube leads to margin violation.

2. For $\xi > 1$ point is misclassified.

3. For $\xi = 0$ point is support vector

ξ is the limit of misclassification permitted in SVM. Thus, the objective function of SVM can be further optimized by granting only ξ_i possible errors. Now, soft-margin SVM will be a bounded optimization objective over w and ξ and is given in eq. (1.6) as:

$$f(x) = \mathbf{w} \cdot \mathbf{x}_i + b \tag{1.6}$$

the constraint $y_i\left(\mathbf{w} \cdot \mathbf{x}_i + b\right) \geq 1 - \xi - i$ can be written more concisely as $y_i f(x_i) \geq 1 - \xi - i$, together with $\xi_i \geq 0$, is equivalent to Equation (1.7):

$$\xi_i = \max\left(0, 1 - y_i f\left(\mathrm{x}_i\right)\right) \tag{1.7}$$

The optimization function can be given in Equation (1.8) as:

$$\underset{w}{min} \quad \underbrace{||\mathbf{w}||^2}_{\text{regularization}} + C \sum_{i=1}^{n} \underbrace{max(0, 1 - y_i f(x_i))}_{\text{loss function}} \tag{1.8}$$

Based on Equation (1.8), data points can be distinguished as:

1. If $y_i f(x_i) > 1$, point is on the correct side of the margin and no penalty for error.
2. If $y_i f(x_i) = 1$, point is on the decision boundary.
3. If $y_i f(x_i) < 1$, point is exceeding the margin criterion and incurs penalty for errors.

The hole data can be fitted in decision surface if ξ_i is chosen to be maximum. However maximum value of ξ_i leads to pay high cost for errors. Here C is the regularization term which balances among error and margin. The hyper parameters of ξ and C sets the limit on maximal tolerance of error in SVM. Thus, the performance of SVM can be defined by finding the optimal value for these parameters.

1. low value of C enables variables to be easily ignored, leading to large margin.
2. high value of C makes variables hard to ignore, leading to narrow margin.
3. $C = \infty$ enforces all variables leading to hard margin.

How do different values of C will effect the SVM classification margin, is shown Figure 1.4. If the value of C is too high, there will be high error cost for misclassified points

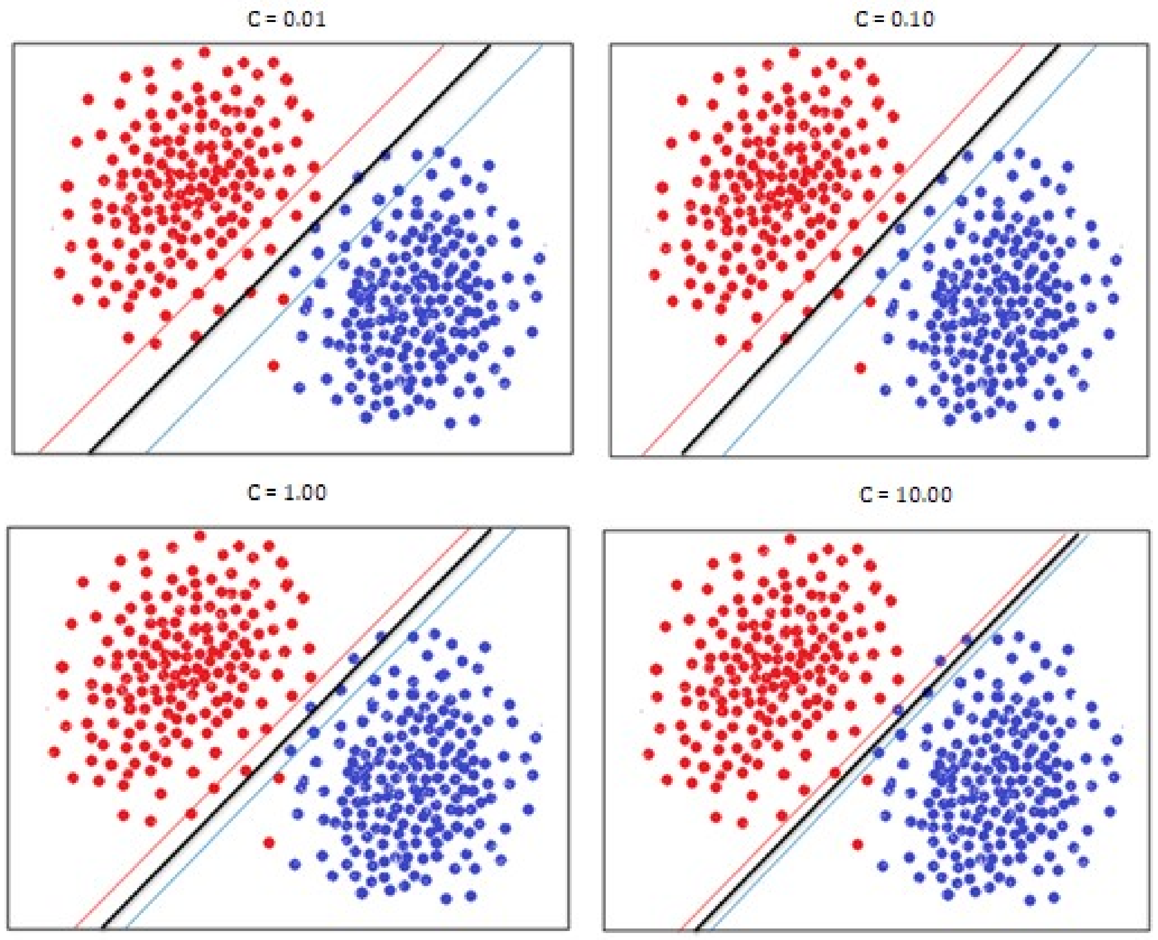


Figure 1.4: Effect of C on soft margin SVM

and required to store more number of support vectors, which may lead to over-fitting. If it is too low leads to model under-fit. Then, the objective function $f\left(x\right)$ represented in given Equation (1.9):

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T \cdot \mathbf{w} + C\sum_{i=1}^{n}\xi_i \quad s.t. \quad y_i\left(\mathbf{w}^T \cdot \mathbf{x}_i + b\right) \geq 1 - \xi, \quad \xi_i \geq 0, \quad i = 1 \ldots n. \tag{1.9}$$

Equation (1.9) is the primal form of soft-margin SVM, the constraints can be easily solved in Dual form of SVM. According to [15], apply Lagrangian function to SVM.

$$\mathcal{L}(w,\xi,b,\alpha) = \frac{1}{2}w^T w + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i\left[1 - \xi_i - y_i\left(wx_i + b\right)\right] + \sum_{i=1}^{n}\alpha_j\left(-\xi_i\right) \tag{1.10}$$

Substituting the derivatives in Equation (1.10), we get:

$=\frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j(x_i\cdot x_j)+C\sum_i \xi_i$

$+\sum_i \alpha_i-\sum_i \alpha_i\xi_i-\sum_i\sum_j \alpha_i\alpha_j y_i y_j(x_i\cdot x_j)-b\sum_i \alpha_i y_i$ -$C\sum_i \xi_i+\sum_i \alpha_i\xi_i$

$=\sum_i \alpha_i-\frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j(x_i\cdot x_j)$

The dual form of soft-margin SVM is:

$$\underset{\alpha}{max} \quad \sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j(x_i\cdot x_j) \quad s.t. \quad 0\le\alpha_i\le C, \quad \sum_{i=1}^{n}\alpha_i y_i=0 \tag{1.11}$$

After applying Lagrangian multipliers, the final objective function $f(x)$ will be the decision function for classification as given in Equation (1.12):

$$sgnf(x)=sgn\left(\sum_{i=1}^{n}\alpha_i y_i K(x_i,x)+b\right) \tag{1.12}$$

Training points which are having α_i values greater than equal to zero are said to be support vectors.

1.2.2 SVM for Regression

SVM is used for both classification as well as regression problems with minor differences in their formulations. Nevertheless, the difference lies in the cost function modelling. The ε-insensitive loss function extended the use of SVM in regression estimation tasks and the technique is known as support vector regression (SVR) [16, 17]. Bothe SVM and SVR utilizes the same theory of optimization but with minor variations in their formulations. In SVR, the dependent variable is having continuous values that allows to produce a curve that can tolerate some error to fit the training data. SVR key idea is to find a curve that minimizes the deviation of the points to the separating hyper-plane. The variants of SVR differ in the way the objective function is formulated and their hyper-parameters are chosen. The regression curve can be constructed based on the number of support vectors produced with respect to the value of ε. High values of ε leads to the selection of few support vectors which indirectly affects the prediction approximation. ε-SVR is the default SVR technique and the other one is ϑ-SVR [15, 18–20]. In ϑ-SVR, a parameter ϑ is added to the original ε-SVR. Unlike in ε-SVR, rather than controlling the maximum allowable ε-deviation, ϑ controls the number of classification errors and the number of support vectors by producing an automatic estimate of ε in the data. $\vartheta \in (0, 1]$, ϑ gives maximum limit on number of classification errors and minimum limit on number of support vectors. ϑ-SVR improves upon ε-SVR by tuning the width of ε-tube automatically to the data and by obtaining bounds on the generalization error [21]. Hence, both C and ε-values will contribute the model complexity. The objective function for SVR is in Equation (1.13):

$$f(x) = w \cdot \phi(x) + b \tag{1.13}$$

The quadratic optimization problem of SVR is given in Equation (1.14):

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} \quad s.t. \begin{cases} y_i - (\mathbf{w} \cdot \phi(\mathbf{x_i}) + b) \leq \varepsilon \\ (\mathbf{w} \cdot \phi(\mathbf{x_i}) + b) - y_i \leq \varepsilon \end{cases} \tag{1.14}$$

w is the weight vector and $\phi(x_i)$ is the dot product of kernel matrix, $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$. $f(x)$ is an optimization problem that uses one slack variable ξ_i in classification to approximate the number of misclassified samples and a pair of slack variables ξ_i and ξ_i^* in regression to approximate the variation among the estimated output and actual output. The optimization function $f(x)$ is solvable by assuming that each data sample (x_i, y_i) of training data T is fitted in the input feature space with an ε-accuracy. Therefore, the objective function $f(x)$ is expressed as a minimization function over w and is given in Equation (1.15):

$$\min_{w,\xi,\xi^*} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \tag{1.15}$$

s.t.
$$\begin{aligned} & y_i - (\mathbf{w} \cdot \phi(x_i) + b) \leq \varepsilon + \xi_i, \\ & (\mathbf{w} \cdot \phi(x_i) + b) - y_i \leq \varepsilon + \xi_i^*, \\ & \xi_i, \xi_i^* \geq 0, \; i = 1, 2, \cdots, n. \end{aligned}$$

Here C is the regularization parameter which balances the classification error and width of the ε-tube. The objective function incurs a cost to the sample whose estimated value is significantly deviated from $f(x)$ by a maximum of ε. The final objective function $f(x)$ will be the following decision function for regression as in Equation (1.16).

$$f(x) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) K(x_i, x) + b \tag{1.16}$$

α_i and α_i^* are lagrange variables and the training points which are having α_i values grater than equal to zero are called the support vectors.

1.2.3 SVM kernels

A kernel is a special kind of similarity function expressed in terms of a dot product. It takes two points as input, and returns their similarity as output. Kernel functions play an important role in SVM and is used to analyze some non-linear patterns in the given dataset by using a linear classifier. SVM algorithm uses kernel-trick for projecting non-linear data and transforming it to linearly separable to create an optimal decision boundary. Kernel SVM handles high dimensional data in a very efficient manner. Thus, SVM extends the class of decision functions to the non-linear case by using kernel trick. With the help of mapping function ϕ, the data X from the input space is transformed to high dimensional feature space, say χ and solves the linear learning problem in χ. Mapping function $\phi : X \rightarrow \chi; \phi(x_i)^T \cdot \phi(x_j)$. While working in higher dimensions is beneficial, it also increases the running time of SVM because of the dot product computation. The mapping function ϕ is induced by a kernel K which calculates the inner product between two points in the feature space, $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$ can reduce the running time. kernel function can be interpreted as a measure of resemblence among data samples (x_i, x_j). Popular SVM Kernel functions are Linear, Polynomial, Sigmoid, Radial Basis function (RBF) and Logistic Kernels. How input data is projected into higher dimensions using Kernel trick in SVM is shown in Figure 1.5.

SVM using linear kernel is the basic and faster learning model used when data is linearly separable. Linear Kernel : $K(x_i, x_j) = (x_i \cdot x_j)$, where x_i and x_j are the data

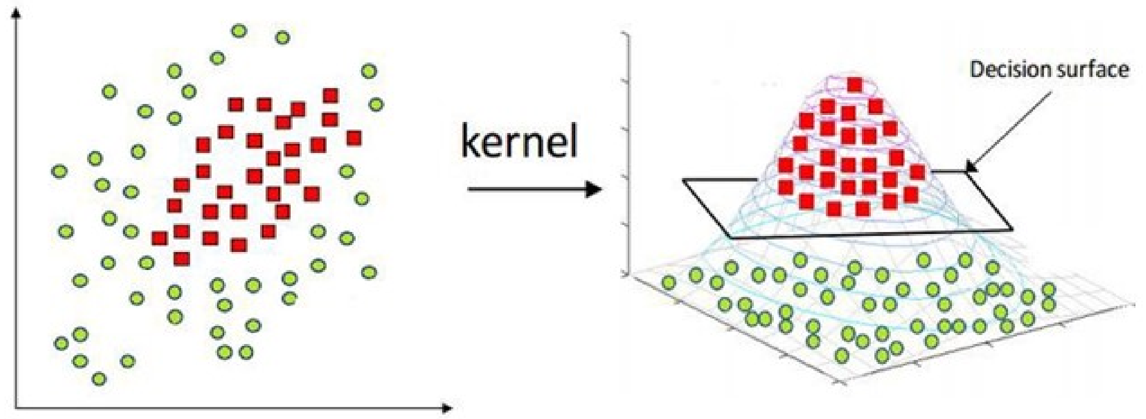


Figure 1.5: Kernel trick in SVM

points to classify. Polynomial Kernel defines the coincidence of original vectors in a input space to the polynomials of vectors used in kernel space. Polynomial Kernel: $K(x_i, x_j) = ((x_i \cdot x_j) + 1)^d$, here d is the degree of polynomial. Sigmoid Kernel takes the form of $K(x_i, x_j) = tanh(\alpha(x_i \cdot x_j) + c)$. There are two adjustable parameters in this kernel, the slope value α and the biased constant c, for some $k > 0$ and $c < 0$. A common value for alpha is $\frac{1}{m}$, where m is the data dimension. RBF kernel is used when there is no prior knowledge on data and the similarity of two samples is judged by their euclidean distance. RBF Kernel: $K(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2})$. Where $\|x_i - x_j\|^2$ is the squared Euclidean distance between two feature vectors (x_i, x_j). σ is a scalar quantity that tells the influence of a data sample in the kernel space. High values of σ typically produce highly flexible decision boundaries, and low values of σ results more linear decision boundary. Logistic kernel is modelled as: $p(y = 1 \mid x) = \frac{1}{1+\exp\left(-w^\top \phi(x)\right)}$ now, the SVM form of w is: $p(y = 1 \mid x) = \frac{1}{1+\exp((-\sum_{i=1}^{n} \alpha_i y_i K(x_i, x_j))}$. p is the posterior probability returned by model. This gives SVM with kernelized logistic model.

A linear SVM works well when data can be linearly separable and considered to be efficient technique in terms of generalization ability and computation time. SVM works fine for low sample and high dimensional data as it uses a portion of training data in the objective optimization (called support vectors), and training time count on number of samples in training data.

However, non-linear SVM classifier is not suitable for time- series datasets as the the number of iterations scales up with the number of training samples that causes slow convergence rate and longer training time (i.e expensive computational complexity). SVM does not perform very well when the datasets has more noise i.e., target classes are overlapping, having uncertain data or outliers. The case where the number of attributes are excessive for each data samples than the number of training samples, data piling arises and SVM will give poor generalization. SVM does not directly provide probability estimates, i.e., probabilistic explanation for the classification of data points is missing.

1.3 Incremental/Decremental SVM

Most of the ML algorithms including SVM, implement batch learning procedures. Batch learning algorithms trains input data which is priory available, and can not handle the sheer volume of continuous data in given time stamp. Batch learning models do not continuously integrate new data into already constructed models. Instead, regularly reconstruct new models from scratch by keeping the system weights constant while computing the error associated with each input sample. This process takes long training time and also leads to potentially intensive computational models. The conventional SVM algorithm does not scale well enough in accordance with time series datasets as the learning system update time will multiply non-linearly along with the length of dataset [22, 23]. Thus, a learning paradigm is required that can effectively process sequential data in a streaming fashion.

The search for SVM involves selecting kernel k and penalty parameter c repeatedly to solve optimization objective. Generally, the search involves tuning of hyper parameters (c, σ) that maximizes its classification ability. The solution of SVM classifier is produced by optimizing the quadratic programming problem (QPP) at each training step. This QPP is an optimization function comprised of number of approximations with respect to the size of training data. SVM performance is tend to deteriorate while processing continuous time-series data as the updations are costly in the context of space and time. The memory requirements and training time makes SVM infeasible and computationally prohibited as the computational complexity can reach upto $\mathcal{O}(n^3)$. Thus the application of kernel SVM to continuous time-series data is challenging. Hence, batch learning can be replaced with incremental learning and can make each iteration very cheap.

Incremental and decremental training of SVM continues with the migration of vectors in and out of the support set along side modifying the associated thresholds. In general, incremental learning of SVM searches for a solution of the $(n+1)$ new training data from an approximate solution of n previous training data and new sample c. To minimize the computational overheads, when new sample point c is available, SVM integrates this new sample into the objective function and modifies the regularization term C and σ accordingly. This incremental approach can also be adjustable for decremental strategy. The decremental unlearning approach uses a leave-one-out (LOO) procedure to estimate the accuracy for each unlearned data sample in the training set. The analysis with the same procedures and with very slight variations, can be used for both the incremental and decremental learning as shown in Figure 1.6. Incremental training of SVM learns new data sample (x_c, y_c) one at time by adiabatically adding to the solution and checking whether KKT conditions are satisfied on all previously learned data. It implements leave-one-out procedure to construct the solution recursively while neglecting the already learnt data leaving the support vectors behind.

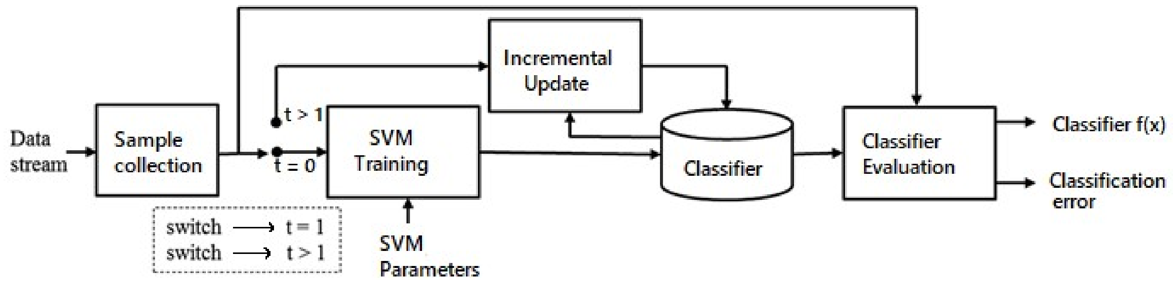


Figure 1.6: Conceptional model of Incremental SVM

However, the performance of incremental/decremental training of SVM can deteriorate by computational overheads and inaccurate results, mainly for a non-linear SVM classifiers with continuous arrival of data samples. The drawback of incremental learning is its slow convergence rate. The convergence rate of SVM can be accelerated by iterative procedures such as Frank-Wolfe (FW) method [24–26].

1.4 Frank-Wolfe Algorithm

SVM solution is generally formulated as a complex quadratic optimization problem (QP) which takes the order of $\mathcal{O}(n^2)$ memory complexity and $\mathcal{O}(n^3)$ time complexity for naive implementation on the training size of n [12, 15, 18]. Further research studies on SVM mainly focused towards the enhancement of SVM for continuous time-series data [27, 28]. Efficient methods are devised gradually to improve the rate of convergence of SVM algorithm. One such method is learning classifier by adopting the concept of transforming data to the solution of processing a minimal enclosing ball (MEB) as shown in Figure 1.7. MEB solution gives a slightly different penalty parameter and lower constraints on the objective function of SVM [29]. Thus efficient algorithms have been devised under the concept of Core Vector Machines (CVMs) to train SVMs. These algorithms are capable of approximating $\mathcal{O}(\frac{1}{\varepsilon})$ iterations with any degree of accuracy ε that is insensitive to the size and dimensions of training data in which the ball is constructed. There are several iterative algorithms are proposed for CVMs further to approximate the optimization of MEB problem [30–32]. SVMs are sparser in terms of training patterns and the model is characterized by a subsample of the original training dataset only. Lately, there has been an interest towards the development of sparse greedy approximation algorithm in ML research is resumed. The recent visit of the solution of MEB problem is termed to be Frank–Wolfe (FW) method [25, 32, 33].

FW algorithm is a first order iterative optimization algorithm developed for solving quadratic programming problems with linear constraints [34]. Recently, it is revisited and modified such that it can be applicable to nonlinear problems with constrained convex optimization functions such as SVM [24, 25, 32, 33, 35]. The naïve strategies of FW algorithm brings the solution in the direction of increase in Optimization objective at each iteration

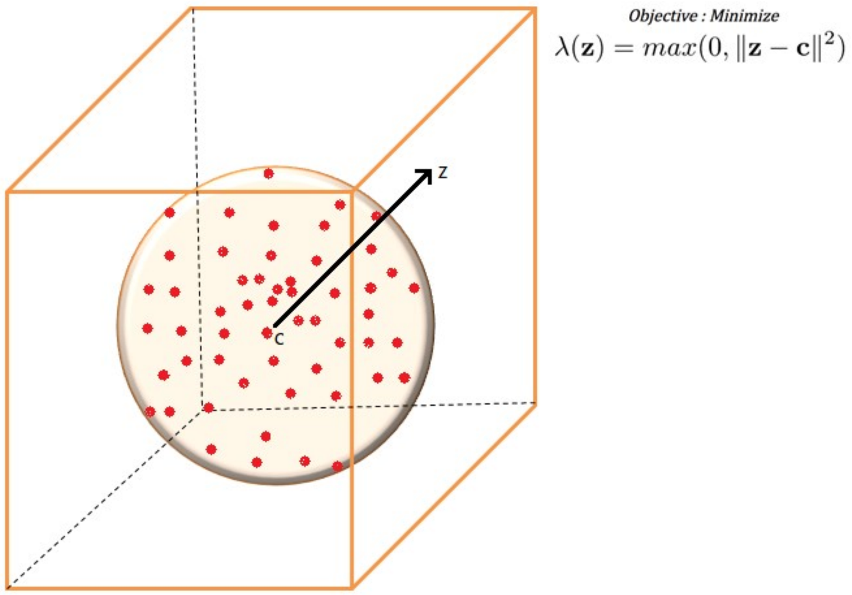

Objective : Minimize
$\lambda(\mathbf{z}) = max(0, \|\mathbf{z} - \mathbf{c}\|^2)$
z
c

Figure 1.7: Minimum Enclosing Ball (MEB)

(Towards step) [32, 34]. The convergence rate of algorithm is pretty slow using only this Towards step. Later, it was modified by introducing an Away-step to boost the convergence rate where the solution moves against to the direction of decrease in optimization objective at each iteration (Away-step) [32]. The choice between these two steps are made at each iteration based on the optimistic path of the possible feature space. The concept of FW algorithm can be integrated with SVM to describe the sparsity and the convergence nature of objective function of SVM [32, 33, 35, 36].

However, the procedure of using Toward step or Away-step of FW algorithm for non-linear SVMs sometimes tend to influence the weight vectors which are not contributing to the optimal solution. This causes considerable deviation of the current approximation solution that results in performance degradation.

1.5 Large Margin Distribution (LDM)

Maximum margin is a fundamental issue of SVMs. SVM considers a single point margin optimization. The function $f(x)$ will not change by adding a new sample (x_i, y_i) as long as $f(x_i)$ does not deviate more than ε margin from y_i, moreover, deviations are penalized. If the scattering of the data in ε -insensitivity zone is drifted utmost from the orientation of support vectors, then the resulting decision boundary will not be effective. ε-SVR and ϑ-SVR are not extremely resistant to outliers. The training points which resides inside of the ε -insensitivity zone completely ignored by the fitting curve of SVR. This concept causes the sparsity nature for SVM but does not force to lower the spread of data samples inside the ε-insensitivity zone. The performance of SVR can be improved by imposing sparsity and minimum disperse of training data simultaneously within ε-insensitivity zone.

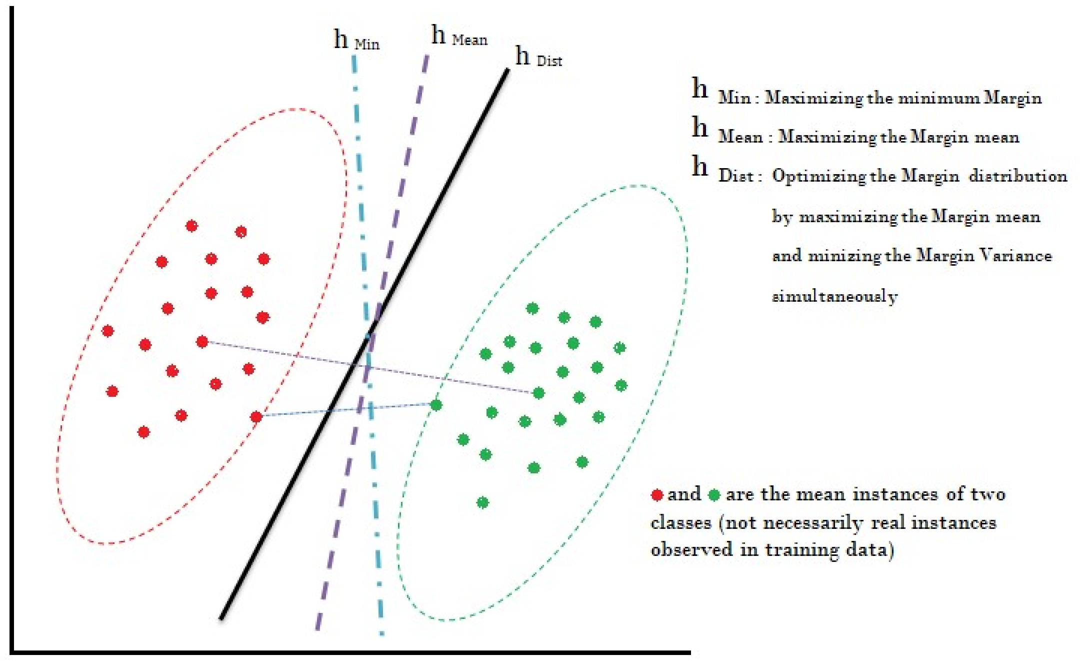

h Min
h Mean
h Dist
h Min : Maximizing the minimum Margin
h Mean : Maximizing the Margin mean
h Dist : Optimizing the Margin distribution by maximizing the Margin mean and minizing the Margin Variance simultaneously
● and ● are the mean instances of two classes (not necessarily real instances observed in training data)

Figure 1.8: Margin Distribution theory in SVM

The modern research disclosed that defining a margin by considering the whole data distribution can improve the performance of SVR. This type of margin, based on data distribution instead of margin which is based on single data sample offers a promising way to address outliers and there by improving the generalization performance of SVR as shown in Figure 1.8. Thus, the margin distribution results good performance in SVR by increasing the mean of the margin and decreasing its residual errors simultaneously [37, 38]. Thus, large margin distribution (LDM) strategy proposed recently can improve the generalization performance of SVM. The learning rate of kernel SVM which uses LDM concept can be speed up by employing coordinate descent (CD) technique [37].

1.6 Distance Weighted Discrimination (DWD)

Incremental learning of kernelized SVM suffers with the curse of kernelization [39]. When data is projected to high dimensional space, it is often that the data is densely distributed on the boundary. Support vector machines used for regression (SVR) is also sensitive to the distribution of boundary data. In terms of data on decision curves (Support vectors), both the variants of SVR, ε-SVR and ϑ-SVR are vulnerable. These countless data vectors in high dimensional space tend to pile up at the decision boundary and treated as Support vectors. If the size of support vectors increases, consequently there will be outgrowth in update time at each iteration of the model [40].

During learning, the optimization function critically depends on width of the margin and this influences the final estimation objective count on the number of the support vectors. SVM can be defined as a optimization approach trying to increase the smallest distance from the samples to the classification boundary over training data. The margin theory i.e., maximizing the minimum margin gives a good encouragement to the accuracy of SVM. Recent advancement in SVM theory, however, revealed that aiming to maximize the minimum distance between instances to the boundary does not always lead to reduced generalized error. But increasing the average distance of the data samples to the decision boundary can improve the generalization performance effectively. [41, 42].

An optimization method called Distance-weighted Discrimination (DWD) technique is proposed as an alternative to address data piling at the margin in SVM [41, 43]. DWD strategy uses an interior point method, which tries to improve the margin mean with respect to whole data and defines a decision curve [44]. DWD takes all data into consideration but gives more significance to those closer to the hyperplane. while DWD overcomes the data-piling and mitigates the overfitting effect, it is sensitive to class imbalance [45]. When the sample size of one class is much greater than the other one, the classification boundary would be pushed towards the minority class and consequently, all future data vectors will be classified into the majority class [46]. Distance-weighted Support Vector Machine method possesses the merits of both SVM and the DWD and can alleviate the data-piling and overfitting problems and also allows faster training approach for large scale datasets [42, 47, 48].

The key idea behind SVM is to find a linear discriminant function $f(x) = w.x + b = 0$ with a direction weight vector w and bias b. Where the data x is assigned to the class $+1$ when $f(x) > 0$ and assigned to -1 when $f(x) < 0$ and also keeps x possibly far away from the decision surface $f(x) = 0$. This is an optimization problem characterised by points nearer to decision boundary. This decision boundary separates the input data space into different regions whose orientation is determined by using coefficient direction vector w and position is identified by intercept b. Data piling happens when plenty of training samples have same projections in the direction of w and all stack up at the same direction of decision boundary. Consider a unit simplex, where two classes of high dimensional low sample size datasets are projected as shown in Figure 1.9. Since there are many zeros in the class -1 vectors(green), they are locate diverse at the vertices of this simplex while class +1(red) are nearer to the center.

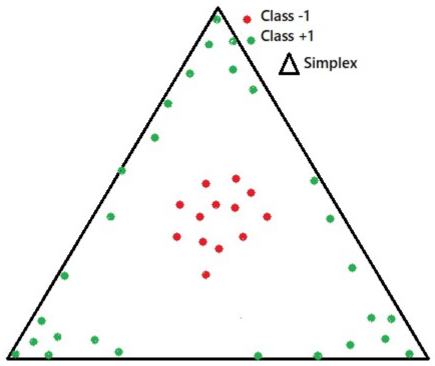


Figure 1.9: Distribution of data vectors on Unit simplex

A classic linear discriminator such as SVM struggles to capture the differentiation between these two classes as it encounters data piling issue. Data-piling occurs when projections of weight vector w generated by a classifier have identical directions and incident on to points as shown in Figure 1.10. Data-piling reflects severe overfitting in the SVM and is an indicator that the direction is driven by noise in the data, and hence the direction as well as the classification performance can be stochastically volatile. DWD was originally proposed to control the data-piling effect but diminishes at overfitting issue as it is subtle to skewed data [49]. In particular, when the sample size of one class is much greater than the other one, the classification boundary would be pushed towards the minority class and consequently, all future data vectors will be classified into the majority class as shown in Figure 1.11.

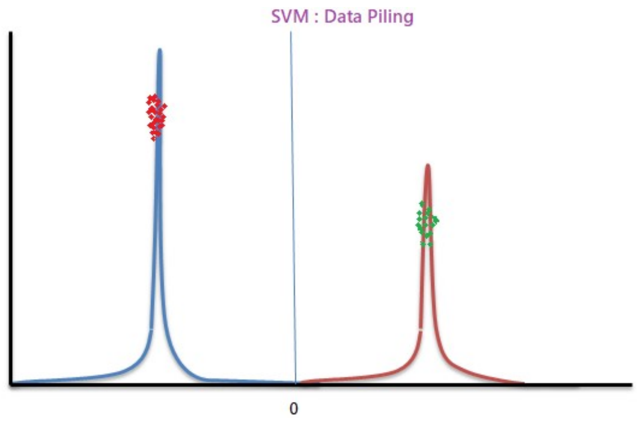


Figure 1.10: Data piling in SVM

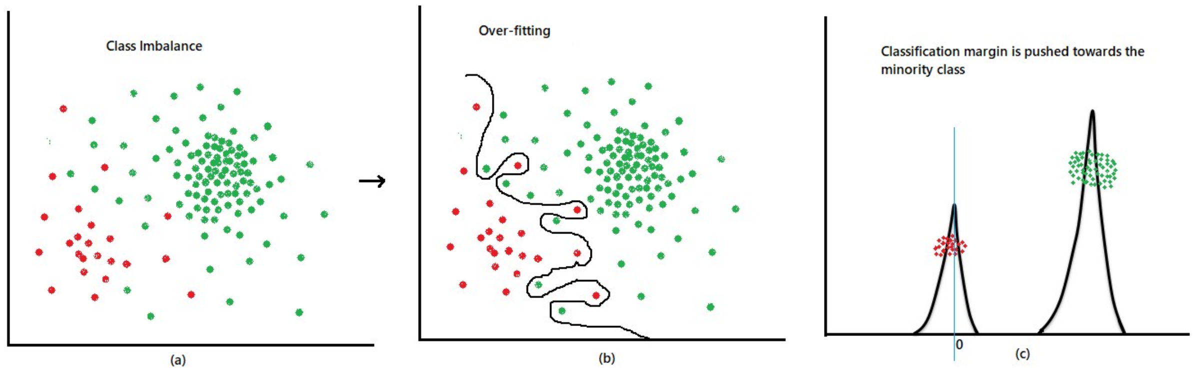


Figure 1.11: Over-fitting issue in DWD

The formulations of SVM objective function which is based on weighted distance will try to reduce the DWD loss and finds the optimal orientation of hyperplane [42, 47].. SVM is evaluated using a minmax optimization formulation, focusing mainly on number of support vectors i.e., samples that resides right on the separating hyper-plane. Where as DWD allows more number of training samples to have direct influence on weight vector w and gives high priority to the training instances that are close to hyper-plane. Distance-weighted Support Vector Machine method possesses the merits of both SVM and the DWD and leads to faster training approach for large scale datasets [42, 47, 48].

1.7 Research direction

Research direction seeking for the possible improvements to the existing optimization techniques which can further improve the performance of SVM is discussed further in following sections.

1.7.1 Problem statement

- For the analysis of continuous time-series data, it is generally anticipated to induce a balance among computational complexity and generalization performance of approximate techniques such as SVM. Non-linear SVMs are computationally prohibitive when used with continuous time-series data as it requires high space and time complexities. The classical SVM is not adjusted to process large training set as the computational complexity can reach $\mathcal{O}(n^3)$. Therefore, non-linear SVM classifier deteriorates to continuous time-series data Thus, the application of nonlinear SVM classifiers to continuous time-series data is challenging. Hence, batch learning can be replaced with incremental learning and can make each iteration very cheap. However, the drawback of incremental learning is its slow convergence rate. The convergence rate of SVM can be accelerated by iterative procedures such as FW method [24–26]. FW algorithm is a first order iterative optimization algorithm developed for solving quadratic programming problems with linear constraints [34]. Recently, it is revisited and modified such that it can be applicable to nonlinear programming problems with constrained convex optimization functions such as SVM [24, 25, 32, 33, 35]. The basic idea behind FW algorithm is, it considers a linear approximation of the objective function and each iteration moves the solution towards a minimizer of this linear function.

- Maximum margin is a fundamental issue of SVMs. SVM considers a single point margin optimization. The function $f(x)$ will not change by adding a new sample (x_i, y_i) as long as $f(x_i)$ does not deviate more than ε margin from y_i, moreover, deviations are penalized. If the scattering of data in ε-insensitivity zone is drifted utmost from the orientation of the support vectors, then the resulting decision boundary will not be effective. ε-SVR is not very robust to the outliers. The training points which resides inside of the ε -insensitivity zone completely ignored by the fitting curve of SVR. This concept causes the sparsity nature for SVM but does not force to lower the spread of data samples inside the ε-insensitivity zone. The performance of SVR can be improved by imposing sparsity and minimum disperse of training data simultaneously within ε-insensitivity zone. Therefore, the proposed work will have a balance among sparsity and minimum distribution of training data in ε-insensitivity zone. To do so, the proposed model utilizes the whole training dataset and avoids overfitting. The modern research disclosed that defining a margin by considering the whole data distribution can improve the performance of SVR. Margin distribution is optimized by increasing the mean of the margin and decreasing its residual errors simultaneously. Thus, the LDM is proved to be essential to boost the generalization performance. The learning rate of kernel SVM which uses LDM concept can be speed up by employing coordinate descent technique.

- Incremental learning of kernelized SVR suffers with the curse of kernelization and cannot handle uncertain data. When data is projected to high dimensional space, it is often that data is densely distributed on the boundary. When it comes to data on decision curves, both SVR variations ε-SVR and ϑ-SVR are susceptible. These countless data vectors in high dimensional space tend to pile up at the decision boundary and treated as Support vectors. When training sets with mismatched sizes are used, the resulting decision boundary unfavourably skewed towards the majority class. If the size of support vectors increases, consequently there will be outgrowth in update time at each iteration of the model [40]. During learning, the optimization function critically depends on width of the margin and this influences the final estimation function count on the scattering of the support vectors. Inspired by the idea of distance weighted strategy, we can adjust the bounds on support vectors of ϑ-SVR that allows the flexibility of specifying errors for uncertain data, thus improving the performance of ϑ-SVR on uncertain data.

- In general, incremental learning and decremental unlearning algorithm of SVM searches for a solution of $(n+1)^{th}$ new training data from an approximate solution of n^{th} previous data and new sample c. This incremental approach is also adjustable for decremental learning strategy. The decremental unlearning approach uses a leave-one-out (LOO) procedure to evaluate the accuracy for each unlearned data sample in the training set. However, the performance of incremental/decremental training of SVM deteriorate by intensive iteration overhead and inaccurate results mainly for a non-linear SVMs with imbalanced datasets. Over the years, several techniques were introduced to boost the performance of SVM to solve skewed vector space problem [50, 51]. These boosting techniques are either classifier independent data driven approach or classifier based cost sensitive approache or combination of these two. The recent literature suggests incremental SVM can be optimized by several boosting techniques to improve the overall performance [52].

The above mentioned challenges and literature motivate the present research work towards enhancing the performance of Support vector machines for generalization and prediction estimation problems using new optimization techniques.

1.7.2 Aim

The classical SVM formulation is limited by the problems that are listed above. The aim of this research is to improve the performance of classical SVMs by introducing new optimization strategies that minimise computational complexity while also improving generalisation ability. We introduce a modified frank-Wolfe algorithm in each iteration of SVM training to speed up the convergence rate on incremental learning. A margin distribution strategy is applied to the SVR problems to address outliers issue. LDM is implemented by modified DCD technique to generate more smoother curve to fit all training data. To address overfitting issue of imbalanced datasets in SVM, we allowed the flexibilities of providing a limit on support vectors for each class labels. This flexibility allows to specify variations in error rate for different class labels, there by improving the performance. To boost up the learning rate of incremental/decremental SVM, we combined weak SVMs with asymmetric misclassification cost to modify the training datasets. This modified dataset is used to boost the prediction accuracy of weak SVMs at each iteration. Later, the accuracies from all of these classifiers are integrated by a weighted majority vote to produce the final class label.

The results show that the new formulations and algorithms meet the aims of this research, and improve the generalization performance with reduced computational complexity upon the classical SVM algorithms. Therefore, we state that the thesis details the investigation of novel techniques to optimize Support vector machines by maintaining the trade-off between generalization performance and computational complexity.

1.7.3 Problem formulation

The research work can be formulated as:

- Get insight into the state-of-the-art Support vector machines and its variants.
- We understand the existing optimization techniques which can improve the performance of support vector machines for classification and regression problems.
- We propose Support vector machines which is optimized with iterative procedure like modified Frank-Wolfe algorithm to increase the convergence rate on incremental learning along with Accuracy.
- We propose ε-Support vector regression which is optimized with large margin distribution technique using modified coordinate descent strategy to improve the generalization performance.
- We propose fast and robust ϑ-Support vector regression on incremental learning

which is optimized to handle uncertain data by adjusting bounds on support vectors and errors.

- We propose Incremental learning and Decremental unlearning of Support vector machines by speeding up its training process.

- Find the future directions in the field of optimization of Support vector machines which can applied to real world problems.

1.8 Proposed Objectives

The proposed objectives of the Research work are stated as follows:

1. Incremental support vector machines optimized with modified Frank-Wolfe algorithm (SVM-MFW).

2. ε-Support vector regression optimized with Large Margin Distribution using modified coordinate descent strategy (ε-MDSVR).

3. Incremental ϑ-Support vector regression optimized with bounded estimation function to handle noisy datasets.

4. Acceleration of incremental and decremental training of Support vector machines.

1.9 Thesis Organization

Thesis is proposed to have 7 chapters, including an introduction and a concluding chapters. The content of each of these chapters is discussed as follows.

The Introduction chapter provided the necessary background and motivation for the work reported in this thesis. The chapter introduced Machine Learning algorithm and Support vector machines. The advantages of SVM and issues related to SVM which degrades its performance is discussed. Some of the existing optimization techniques are discussed which are used to improve the performance of SVM. At last, research direction seeking for the possible improvements to the existing techniques which can further improve the performance of SVM are proposed.

Chapter-2 reports the recent state-of-the-art techniques to enhance the performance of SVM and discussed the observations and motivations for the research work is to be carried out. Research problem of how SVM can be optimized to improve its performance is also discussed in this chapter.

Chapter-3 presents a novel Support Vector machines optimized with modified Frank-Wolfe algorithm called as SVM-MFW trained by an Incremental learning. When new samples arriving in a streaming pattern, the proposed model learns new sample one by one while maintaining an equilibrium on Karush-Kuhn-Tucker (KKT) conditions. Convergence rate of proposed algorithm is speed up by incorporating modified Frank-Wolfe method (MFW). Modified FW method uses a new “away step” methodology that can boost the convergence rate of algorithm and can quickly reach an acceptable accuracies in the very early iterations. Thus, proposed algorithm is computationally cheaper since it requires simple updations at each iteration to converge to an acceptable accuracy. The model produce sparser representation of support vectors and is more stable with respect to the selection of hyper parameters. Experiments on Parkinson’s disease dataset show that the proposed model (SVM-MFW) gives better results in comparison to the existing algorithms.

Chapter-4 explains about proposed ε-Support vector regression algorithm (ε-MDSVR), that is optimized with the LDM technique by employing modified dual coordinate descendent technique to enhance the performance. The proposed ε-MDSVR model uses the whole training data to define hyperplane to handle over-fitting issue. The model considers the margin based on the whole data distribution to reduce the scattering of data in ε-insensitivity zone to handle outliers. The margin distribution is characterized by mean and its residual error. To achieve better generalization performance, the proposed model maximizes the mean and minimizes the residual error of functional margin. Besides, to increase the learning rate, the proposed model is integrated with modified Dual coordinate descent (DCD) strategy. The modified DCD method orderly updates one variable at a time. The variable is selected for updation such that if it possibly derives the maximum optimization in objective function value. This strategy can increase the learning speed and improve the generalization performance. The proposed method is implemented using some popular datasets taken from the UCI machine learning repository. The proposed algorithm tries to generate more smoother regression curve and achieves better prediction accuracy in less computation time compared to classical SVR.

Chapter-5 details the proposed ϑ-Support vector regression algorithm (ϑ-PSVR), that is used to handle uncertain data and distribution of data on boundary by applying bounded functions to adjust the number of support vectors and errors. The proposed method formulates maximum and minimum limit functions on perturbed data. Two special adjustments to the support vectors and penalty parameters are used to enable the model to learn incrementally. The projections of data points to the fitting surface are generated once the value of w is computed. The proposed method uses ϑ-SVR trained over incremental learning and optimized to handle noisy data to enhance the performance. Experiments on Parkinson's disease data taken from PPMI repository show that the algorithm tries to generate a more smoother regression curve and achieves better prediction accuracy compared to classical ϑ-SVR.

Chapter-6 explains about boosting algorithm to enhance the performance of incremental learning and decremental unlearning of support vector machines used on imbalanced datasets. In case of imbalanced data, combining weak SVM classifiers can make a prediction better. The proposed boosting algorithm combines weak SVMs with asymmetric misclassification cost to modify the training datasets. This modified dataset is used to boost the prediction accuracy of weak SVMs at each iteration during training. Later, the predictions from all of these classifiers are then combined by a weighted majority vote to produce the final class label. The proposed method is applied on some synthetic dataset. The algorithm is validated on number of support vectors, error vectors, trajectory of coefficients α_i, trajectory of LOO margin g_i, iterations and training time.

The conclusions of the dissertation and future directions are outlined in chapter 7. SVM is a margin-based supervised machine learning technique which can be applied on classification and regression estimation tasks. However, non-linear SVM for continuous/large datasets suffer from computational overheads and poor generalization performance. The proposed research work addresses some of the issues like slow convergence rate, data piling, outliers, long training time and poor performance present in SVM when used nonlinear datasets of various shapes, configurations, and distributions. The work proposes optimization techniques to improve the performance of SVM trained over incremental learning that can be used for both classification and regression tasks. As future work, these optimized SVM techniques can be applied to more real world practical datasets.

1.10 Summary

This chapter provided the necessary introduction and back ground formulations for the work reported in this thesis. The advantages of SVM and issues related to SVM which degrades its performance is discussed. Some of the existing optimization techniques are discussed which can improve the performance of SVM. At last, research direction seeking for the possible improvements to the existing techniques which can further improve the performance of SVM is discussed further. The proposed research address some of the issues like slow convergence rate, data piling, outliers, long training time and poor performance present in SVM when used with continuous heterogeneous datasets. The proposed optimization techniques discussed in chapter 3, 4, 5 and 6 can improve the performance of SVM trained over incremental learning and used for classification and prediction problems.

Chapter 2

Literature Survey

In this chapter, a brief survey of the literature related to the contributions made in this thesis is detailed. The chapter is organized as follows: Section 2.1 covers some studies related to SVM and the development of its variants to address some of the issues which effect its performance. Section 2.2 discuss some works related to incremental learning of SVM. Also review the problems encountered while learning non linear SVM over time series data sets and high dimensional low sample datasets. Section 2.3 lists the literature of SVM techniques to scale up with the time series data and also discussed the boosting techniques for fast convergence of SVM. Section 2.4 gives the details of Frank-Wolfe method that can be used to speed up the training of SVM. Section 2.5 and Section 2.6 describes the optimization techniques that can address datapiling, overfitting issues for skewed datasets in SVR. Finally, the summary of this chapter is provided in Section Section 2.7.

2.1 SVM: Learning theory

SVM was first introduced by Cortes and Vapnik (1995) [12] based on statistical learning theory. Scholkopf et al. (1996) [53] represented the SVM decision boundary in terms of a small subset of training data called the support vectors. The authors devised the ε-insensitive loss function, to achieve high generalization ability by minimizing a bound on the expected test error. Burges (1998) [54] described linear SVMs for separable, non-separable datasets and Kernel mapping technique for non-linear SVMs in detail. Also discussed the unique and global solutions of SVM through practical implementation. Joachims (1998,1999) [27, 55] explored the use of SVMs for text classification tasks as well as for large scale datasets. The author showed that SVMs are robust and achieve substantial improvements over the other ML methods, also discussed advances in Kernel SVM. At first SVMs are developed for the pattern classification problems, but later it was extended to solve regression estimation problems as well. Scholkopf et al. (1996) [53] introduced ε-insensitive loss function for the sparseness of SVM to carry over to the case of Regression problems, thus lead to the variant of SVM called ε-SVR. Scholkopf et al. (2000) [15] proposed a new class of SVM for classification and regression by introducing a ϑ which lets to control the number of support vectors. The work of Cortes and Vapnik (1995) [12] revealed that, by maximising the smallest distance between the training samples from the separating hyperplane, the SVM classifier seeks to minimise generalisation error. SVM works on the principle of Margin based theory. Also, The works of Scholkopf (2000) and Hofmann et al. (2008) [15, 16, 56], studied that SVMs are defined by optimizing a regularized risk on the training data, that can give good generalization performance. However, the authors also mentioned that the optimization problem in SVM is usually formulated as complex quadratic optimization problem (QP), for which a basic implementation requires $\mathcal{O}(n^2)$ space and $\mathcal{O}(n^3)$ time complexities when the number of data samples n are large enough, and that are computationally prohibitive. Therefore, using basic numerical approaches to solve QP problem of SVM for datasets with huge volumes and dimensions has been infeasible and computationally prohibitive.

2.2 Incremental learning of SVM

Bradley and Mangasarian (2000) [57] discussed the ability of SVM to handle very large classification problems by dividing the problem into chunks of small linear problems. Authors proved that SVM terminates in a finite number of steps at an exact solution by giving an optimal separating plane for large datasets with huge dimensions. The existing methods for dealing with huge datasets [57] divide the solution space into sub-problems and apply iterative component-wise objective function optimization to these small areas of the original datasets. However, these approaches produce only approximate findings and require numerous scans of the given dataset to get a satisfactory convergence. Besides, SVM quadratic programming solution suffer from high memory requirement and more CPU time when trained in a batch mode on large datasets. Gauwenberghs & Poggio, (2001) [58] stated that the conventional SVM algorithm do not scale well enough in accordance with large datasets as they use batch learning approaches. The learning system update time will multiply non-linearly along with the length of dataset. Incremental training procedures seem to be more powerful in this case as they use gradient estimation of weights to train a learning model when a newsample is added to training data. Diehl and Gauwenberghs, (2003) [59], Liu et al. [60], proposed a combined incremental learning and decremental unlearning approach to train an SVM classifier. The authors supplied the exact solution for the $n + 1$ training data point as a function of the solution of n data and a new data point c. Shilton et al. (2005) [61] discussed that when adding new data to the already trained SVM, batch SVM must be re-trained from scratch. also, adding small set of data to a large training set does not always effect the decision surface. Resolving the problem from scratch seems computationally wasteful. Authors proposed that using the solution of already trained SVM as a starting point to find a new solution. This approach is the heart of active set optimization methods, in fact, an extension to incremental learning. Despite their slower rate of convergence than batch approaches, incremental methods can make each iteration quite inexpensive. However, for a non-linear SVM on big datasets, the incremental/decremental training technique has high computational cost and erroneous outputs. Non linear SVMs suffer with slow convergence rate and long training time also gives poor classification performance.

2.3 Active sets in SVM Learning

Tsang et al. (2005,2006) and Scheinberg et al. (2006) [29, 62, 63] discussed that as the typical dense structure of hessain and constraint matrix of QP involved in optimization objective of SVM, traditional optimization methods seem to be impractical to train an SVM on large scale time series data. This problem is well addressed by an active set method. Active set strategies allows limited number of variables to be updated at each iteration. Fan et al. (2005) [64] stated that for non-linear SVMs, the subset of training examples which are allowed to change are called a working set and proposed the popular Sequential Minimal Optimization (SMO) algorithm that utilizes this concept. However, the works of Platt et al. (1999), Fan et al. (2005), Lee and Huang (2007) [64–66] discussed that At each iteration of SMO, only two variables are chosen for updating. This process generally exhibit a slow convergence rate and poor performance results. Thus, require more efficient methods that can improve the rate of convergence of SVM algorithm.

The common techniques to handle large datasets by SVM is basically to build a solution by solving a sequence of small scale subproblems. But still SVM suffers with low performance issue. Fine and Scheinberg (2001) [67] attempted to scale up SVM methods to large datasets by adapting interior point methods to some classes of the SVM QP problem. Their work is based on the rank of the kernel matrix as a source for further enhancement of SVM and showed that better interior point method can be designed by using a low rank kernel matrix in terms of storage requirements as well as computational complexity. In large-scale SVM problem, however the resulting rank of the kernel matrix can still be too high to be handled efficiently in interior point method. The reformulation of the SVM objective function, the use of sampling methods to reduce the number of variables in objective function of SVM and the combination of small SVMs using ensemble methods have also been explored recently.

Looking for more efficient methods, Tsang et al. (2005) [29] proposed a new approach by adopting the concept of transforming data to the solution of processing a minimal enclosing ball (MEB). SVM can be resembled to this MEB solution such that it gives a slightly different penalty parameter and lower constraints on the objective function of SVM. Yildirim et al. (2008) [31] proposed two iterative algorithms for the task of approximating and optimizing the solution of MEB. Recent advances in computational geometry such as the works of Zhang et al. (2006)[68] have demonstrated that there are algorithms capable of approximating a MEB with any degree of accuracy ε in $\mathcal{O}(\frac{1}{\varepsilon})$ iterations independently of the number of points and the dimensionality of the input space in which the ball is built. Adopting one of these algorithms, Tsang et al. (2006), Lee et al. (2007) and gartner et al. (2009) [36, 63, 66] devised the Core Vector Machine (CVM), and discussed that the model recursively solves the optimization objective by selecting small portion of data for updation. The algorithm seeks for a point outside the approximation of the MEB achieved so far at each iteration. If the point exists, it is combined with the prior subset of data to form an optimization problem for solving the new MEB approximation. The method is repeated until no points outside of the current approximating ball are located within a specified tolerance. CVMs hence require the use of an external numerical solver to tackle a series of increasingly complicated optimization problems.

2.4 Frank-Wolfe optimization technique

Bottou et al. (2004)[69] stated that, it is to be desired to make trade-off between computational complexity and accuracy of SVM for large-scale problems. Inspired by CVMs and greedy approximate solutions, Ouyang et al. (2010) [32, 70] proposed new first order iterative algorithm framework called Frank-Wolfe optimization technique which can be applied to SVM. Frank–Wolfe algorithm (FW) is a first order iterative optimization algorithm developed for solving quadratic programming problems with linear constraints proposed by Frank et al. (1956) [34] It is recently studied in the works of [31, 32]. The MEB and other convex optimization problems, such as SVM, can be approximated using FW. The main concept of FW method is to solve the optimization issue by linearizing the objective function at the current feasible solution and doing an exact line search in the direction acquired from the linearization at each iteration. As a result, each iteration becomes relatively inexpensive and does not necessitate the use of an external numerical solution. This approach finds the best set of weights by incrementally discovering the samples that become support vectors in the SVM model. In addition, the algorithm's rate of convergence is asymptotically linear. Recently, The works of Jaggi et al. (2012,2013), locaste et al. (2013) and Frandi et al. (2013) [24, 25, 33, 35], revisited this FW algorithm and modified such that it can be applicable to nonlinear programming problems with constrained convex optimization functions such as SVM. The naïve strategies of FW algorithm brings the solution in the direction of increase in Optimization objective at each iteration (Towards step) [32, 34]. The convergence rate of algorithm is pretty slow using only this Towards step. Later, it was modified by introducing an Away-step to boost the convergence rate where the solution moves against to the direction of decrease in optimization objective at each iteration (Away-step) [32]. [32, 33, 35, 36].

As stated by the work of Nanculef et al. (2014) [26], the choice between these two steps are made at each iteration based on the optimistic path of the possible feature space. The concept of FW algorithm can be integrated with SVM to describe the sparsity and the convergence nature of objective function of SVM.

2.5 Distance weighted SVM

Wang et al. (2012) [39] discussed that incremental learning of kernelized SVM suffers with the curse of kernelization. Marron et al. (2007) [41] proved that When data is projected to high dimensional space, it is often that the data is densely distributed on the boundary. The authors revealed that there is a substantial data piling of SVM in high dimensional and low sample data contexts. In the works of Xie et al. (2019) [40], authors also discussed the same point as in [41] that support vectors are tend to be very numerous in high dimensional space and all pile up at boundaries of the margin. If the number of support vectors (SVs) increases in each iteration of the optimization, there will be a non-linear growth of model update time and prediction time with data size. To address this issue, Marron et al (2007) [41] proposed a novel discrimination method called Distance weighted discrimination (DWD) as an alternative to SVM. DWD allows more data points to have a direct impact on the objective function and increases the average distance of the functional margin of all data by allowing these distances to influence the separating hyperplane that can optimize the performance. Qiao et al. (2010) [45] developed an optimal weighting scheme of DWD and discussed its advntages and disadvantages. Qiao et al. (2013) [49] investigated two popular large margin classification methods, SVM and DWD under high dimensional low sample size data and the imbalanced data. [42] revealed that aiming to maximize the minimum distance between instances to the boundary does not always lead to reduced generalized error. But increasing the average distance of the functional margin of whole data can improve the generalization performance effectively. Qiao et al. (2015) [46] discussed that when the sample size of one class is much greater than the other one, the classification boundary would be pushed towards the minority class and consequently, all future data vectors will be classified into the majority class. The authors Qiao et al. (2013), Wang et al. (2018) and Lam et al (2018) [42, 47, 48], The algorithm calculates the classification direction by minimising the DWD loss and using SVM to determine the intercept term. As a result, SVM's data-piling and over-fitting issues are no longer an issue. The authors also said that this method is not affected by data imbalance, which was one of SVM's key advantages over DWD. We can improve the original optimization target for our support vector regression issues by incorporating the concept of the average width of the functional margin that characterises margin distribution.

2.6 Large Margin Distribution

SVM is widely recognised as a margin-based learning strategy that aims to optimize the shortest distance between examples and the classification hyperplane. As a result, the margin theory lends support to SVM's generalization performance. Specifically, [71] first suggested margin theory to resist overfitting in some boosting algorithm. After [72] indicated that the minimum margin is crucial and tried to maximize the minimum margin but lead to poor generalization performance. Later, [73] found that although the works of [72] produced larger minimum margin, but suffered from a poor margin distribution. Thus, the authors revealed that apart from minimum margin, margin distribution also plays important role in achieving good generalization performance. Above proposition has been theoretically studied in the work of [74] and proven by [75], Moreover, the works of [37, 38] revealed that, rather than focusing solely on a single point margin, both the margin mean and variance are crucial in acheiving margin distribution. However, all of these theoretical research focused on boosting-style algorithms, while the impact of the margin distribution on SVMs has not been fully explored in practise. Inspired by the idea of Large margin

Distribution Machine, we can optimize the margin distribution by maximizing mean and minimizing variance together with maximum margin to achieve strong generalization performance in SVM.

2.7 Summary

Chapter 2 reports the recent state-of-the-art works on enhancing the performance of SVM. This chapter discussed the observations and motivations towards the present proposed work. The objectives of the present work that is derived from the literature and the observations are also discussed. Research problem of how SVM can be optimized to improve its performance that balances the trade-off between accuracy and computational complexity is also mentioned.

Chapter 3

Incremental support vector machines optimized with modified Frank-Wolfe algorithm (SVM-MFW)

This Chapter describes the novel technique of ensemble SVM trained with incremental learning and optimized with modified Frank-Wolfe algorithm. The chapter introduces the SVM methodology in brief and describes incremental learning by adopting weights associated to each class of the data samples. The issue of slow convergence rate in SVM is solved by incorporating modified Frank-Wolfe algorithm. The effectiveness of SVM-MFW is shown through experiment on Parkinson's disease dataset and the results are summarised. The main contributions of this chapter are given as follows:

- A fast and robust support vector machines trained with incremental learning and incorporated modified Frank-Wolfe algorithm is proposed.
- The algorithm is used for both classification and prediction estimation problems.
- The proposed method uses low iteration complexity and converges to a better solution by producing sparser representation of support vectors.
- The model handles the optimization problem by using a modified form of AWAY-step of Frank-Wolfe algorithm to speed up the training process and converges faster

to acceptable accuracies, thereby reducing the computation time as well.

- The model is experimented on Parkinson's dataset and is evaluated using several SVM kernels. The results are compared to the classical SVM and other ML approaches in terms of accuracy and computation time.
- It is found that the results are much better when compared with existing methods. Thus, the proposed model can be used as a fast and robust classification, prediction tool for PD diagnosis.

3.1 Introduction

For non-linear SVM situations, it is often anticipated to make a trade-off between computational complexity and the accuracy of underlying optimization algorithms. However, SVMs are not able to handle heterogeneous data effectively because they are costly in terms of memory and computing time. The classical SVM is not adjusted to process online data as the computational complexity can reach $\mathcal{O}(n^3)$. Thus, the scalability of nonlinear SVM classifiers to time series problems is challenging. Hence, batch learning can be replaced with incremental learning and can make each iteration very cheap. However, the drawback of incremental learning is its slow convergence rate. The convergence rate of SVM can be accelerated by iterative procedures such as Frank-Wolfe (FW) method [24–26].

The naïve strategies of FW algorithm brings the solution in the direction of increase in Optimization objective at each iteration (Towards step). The convergence rate of algorithm is pretty slow using only this Towards step. Later, it was modified by introducing an Away-step to boost the convergence rate where the solution moves against to the direction of decrease in optimization objective at each iteration (Away-step). The choice between these two steps are made at each iteration based on the optimistic path of the possible feature space. The concept of FW algorithm can be integrated with SVM to describe the sparsity and the convergence nature of objective function of SVM. However, the procedure of using Toward step or Away step of Frank-Wolfe algorithm for non-linear SVM can increase the weights of vectors which do not correspond to optimal solution, causing considerable deviation of the current approximation solution that results in performance degradation. Proposed modified version of FW uses a new ”away-step” technique, which only increases the weights of the vectors corresponding to optimal solution and also eliminates the spurious points. This is achieved by computing the optimal step-size analytically at each iteration. As a result, the computational complexity per iteration is reduced and the number of iterations are independent of the size of the dataset and their attributes.

This chapter presents a novel technique of incremental SVM assimilated with MFW algorithm resulting in a new approach called SVM-MFW. SVM-MFW does not involve any additional feature reduction techniques to speed up the training. Instead, it uses modified FW method to accelerate the convergence rate of algorithm. Thus, making the system more robust to (unseen) data and also resulting in a faster training approach for incremental data. Consequently, SVM-MFW algorithm learns incrementally and is computationally cheaper since it requires fewer iterations to train, and accelerated convergence rate reduces the computational complexity as well. Experimental results show that the proposed SVM-MFW is a fast and stable classification and prediction technique that can improve the generalization performance and speed up the training process.

The rest of the content of this chapter is organized as follows. Section 3.2 presents the basic equations used to derive the proposed model. Formulations to derive the proposed algorithm is described in Section 3.3. Datasets used for the implementation of model is given in Section 3.4. Experimental results and evaluation of the proposed model and the behaviour of the proposed model against other recent existing research is discussed in Section 3.5. The Summary of the chapter is given in Section 3.6.

3.2 Preliminaries

This section explains the basic equations of SVM and Frank-Wolfe algorithms used to derive the proposed model.

3.2.1 Classical SVM

SVM is used for both classification as well as prediction problems with minor differences in their formulations. Nevertheless, the difference lies in the cost function modelling. The objective function for SVM is:

$$f(x) = w \cdot \phi(x) + b \tag{3.1}$$

where $w \in \mathbb{R}^m$ is the weight vector and ϕ is the mapping function induced by a kernel matrix, $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$. The function $f(x)$ is an optimization problem that uses one slack variable ξ_i in classification to approximate the number of misclassified samples and two slack variables ξ_i and ξ_i^* in regression to approximate the variation between the estimated output and the target output. The optimization function $f(x)$ is solvable by assuming that each data sample (x_i, y_i) of training data T is fitted in the input feature space with an acceptable ε accuracy. Then, the objective function $f(x)$ is a constrained minimization problem given for classification in Equation (3.2) and for regression in Equation (3.3):

$$\underset{w,\xi}{min}\frac{1}{2} \parallel w \parallel^2 + C\sum_{i=1}^{n} \xi_i \tag{3.2}$$

s.t. $\quad y_i(w \cdot \phi(x) + b) \geq 1 - \xi_i$

$\qquad \xi_i \geq 0, \; i = 1, 2, \cdots, n.$

$$\underset{w,\xi,\xi^*}{min}\frac{1}{2} \parallel w \parallel^2 + C\sum_{i=1}^{n} (\xi_i + \xi_i^*) \tag{3.3}$$

s.t. $\quad y_i - w \cdot \phi(x_i) - b \leq \varepsilon + \xi_i,$

$\qquad w \cdot \phi(x_i) + b - y_i \leq \varepsilon + \xi_i^*,$

$\qquad \xi_i, \xi_i^* \geq 0, \; i = 1, 2, \cdots, n.$

Here C is the regularization parameter which balances the classification error and width of the ε-tube. The objective function incurs a cost to the sample whose estimated value is significantly deviated from $f(x)$ by a maximum of ε. After applying Lagrangian multipliers, the final objective function $f(x)$ is given as follows:

The decision function for classification:

$$sgnf\left(x\right) = sgn\left(\sum_{i=1}^{n}\alpha_i y_i K\left(x_i, x\right) + b\right) \tag{3.4}$$

The regressive function:

$$f\left(x\right) = \sum_{i=1}^{n}\left(\alpha_i - \alpha_i^*\right) K\left(x_i, x\right) + b \tag{3.5}$$

α_i and α_i^* are Lagrange variables and the samples hich are having α_i and α_i^* greater than or equal to zero are called the support vectors (SVs).

3.2.2 Frank-Wolfe Algorithm

Frank–Wolfe algorithm (FW) is a first order iterative optimization algorithm developed for solving quadratic programming problems with linear constraints [34]. The problem of maximizing a concave function $f(x)$ in the unit simplex Δ can be solved by approximating the solutions $x^{(k)}$, $\forall k = 0 \cdots \infty$ for given K-dimensional face of Δ and the vertices of Δ are the points $e^{(i)}$, $i = 1 \cdots n$ where $e^{(i)}$ has coordinate $e_i^{(i)} = 1$, and all other coordinates are zero such that $f(x^{(k)}) \geq f(x^*) - \mathcal{O}(\frac{1}{k})$. $f(x^*)$ is the maximum value of f in Δ. Let x^* be the vertex of Δ with maximum value of f. Approximate solutions $x^{(0)}, x^{(1)}, \cdots x^{(k)}$. $\forall k = 0$ to ∞ are obtained by finding the index i^* of the largest coordinate of the gradient $\bigtriangledown f(x^{(k)}$. Let $x^{(k+1)}$ is the point on the segment from $x^{(k)}$ to the $e^{(i*)}$ that maximizes $f(x)$ as shown in Figure 3.1. Then FW finds ($\alpha^* \in [0, 1]$) that maximizes $f(x^{(k)} + \alpha(e^{(i*)} - x^{(k)}))$, where $x^{(k+1)} = x^{(k)} + \alpha(e^{(i*)} - x^{(k)})$. The procedure optimizes the function by selecting only largest component of the gradient rather than optimizing in the direction of the gradient, so that $x^{(k)}$ has at most $k + 1$ nonzero entries. Also, the search direction $e^{(i*)} - x^{(k)}$ is used, from $x^{(k)}$ toward $e^{(i*)}$, but not the direction $e^{(i*)}$; this keeps the search within Δ.

The Frank-Wolfe approach is intended for large-scale datasets of the Equation (3.6), which involves optimizing a linear function $f(x)$ across a working set S.

$$\underset{x}{max} \quad f(x) \quad s.t. \quad x \in S = \{x \in \mathbb{R}^m : \sum_i x_i = 1, x_i \geq 0\}. \tag{3.6}$$

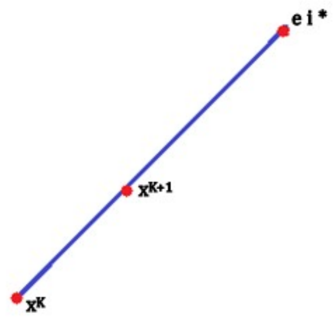


Figure 3.1: Unit simplex Δ vectors

It generates a sequence of approximations $\{x_0, x_1, \cdots, x_k\}$ to a solution of above problem at every iteration k by finding a feasible search direction with an optimal step-size. Thus, the x_k has a sparse representation, which makes the algorithm suitable even for high dimensional data. This problem can also be represented by popular ML algorithm such as SVM [26]. The SVM optimization that matches the FW approach in Equation (3.6) is as follows:

$$\underset{x}{max} \quad f(x) = -x^T \mathcal{K} x \quad s.t \quad \sum_i x_i = 1, x_i \geq 0, \forall i = 1, \cdots, k. \tag{3.7}$$

Where $\mathcal{K}$ is the same kernel function used in SVM. The algorithm generates $\{x_{(k)}\}$ at each iteration k by computing optimal step-size ($\alpha \in [0, 1]$) analytically until it converges to an acceptable accuracy ε. At each iteration t, FW performs a search for finding a vertex $e^{(i*)}$ such that it maximizes the $f(x)$ at x_t. Let $x_{(k+1)}$ be the point on the segment from $x_{(k)}$ to the $e^{(i*)}$ that maximizes $f(x)$ as shown in Figure 3.2, it finds a step size ($\alpha^* \in [0, 1]$) that maximizes $f(x_{(k)} + \alpha(e^{(i*)} - x_{(k)}))$ and moves the function in same direction.By using a step size ($\alpha^* \in [0, 1]$) it computes $x_{(k+1)}$ = $x_{(k)} + \alpha(e^{(i*)} - x_{(k)})$.

The main drawback of using only this Toward step is its poor convergence rate, which is pretty slow when the descent directions $(e^{(i*)} - x^{(k)}))$ are not adequate. So the use of

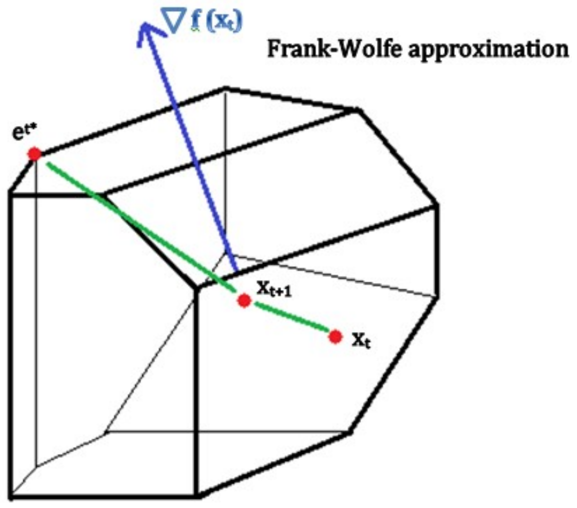


Figure 3.2: Frank-Wolfe approximations in K-dimensional space of Δ

Away-step in FW algorithm has been introduced [32]. To boost the convergence rate of FW, each iteration can move the solution away from the direction in which the objective function decreases (Away step) [32]. At each iteration t, FW performs a search for finding a vertex $e^{(j*)}$ such that it minimizes the $f(x)$ at x_t.

Let $x_{(k+1)}$ be the point on the segment from $x_{(k)}$ to the $e^{(j*)}$ that minimizes $f(x)$ as shown in Figure 3.2. Away-step finds a step size ($\alpha^* \in [0,1]$) that minimizes $f(x_{(k)} + \alpha(x_{(k)} - e^{(j*)}))$ and moves the function in that direction. By using a step size ($\alpha^* \in [0,1]$) it computes $x_{(k+1)} = x_{(k)} + \alpha(x_{(k)}) - e^{(j*)}$. The choice between these two steps are made at each iteration based on the promising direction of the feasible vector space. This method has been successfully used to train non-linear Support Vector Machines on a large-scale datasets. Specializing FW to SVM training has allowed to obtain efficient algorithm, and also important theoretical results, including convergence analysis of training algorithms and new characterizations of model sparsity.

3.3 Proposed SVM-MFW

This section first devises modified form of SVM to address class imbalance problem and gives the formulations for incremental learning and then explains the integration of modified Frank-Wolfe algorithm. The terminology used in proposed work is: Let T be the training set of n samples of (x_i, y_i) with x_i input values and the binary class label of $y_i \in (-1, +1)$.

3.3.1 Modified SVM for class imbalance data

To avoid increased risk of overfitting for class imbalance issue in SVM, we give equal weight to both the positive and the negative classes. In case of classification, a penalty parameter (C^{y_i}) value for each class of data is used as given in Equation (3.8).

$$\underset{w,\xi}{min} \quad \frac{1}{2}\| \ w \ \|_2^2 + C^{(-1)} \sum_{y_i=-1} \xi_i + C^{(+1)} \sum_{y_i=+1} \xi_i \tag{3.8}$$

$$\begin{aligned} \text{s.t} \quad & y_i \left(w \cdot \phi\left(x\right) + b \right) \geq 1 - \xi_i \\ & \xi_i \geq 0, \ \ i = 1, 2, \cdots, n. \end{aligned}$$

After applying Lagrangian multiplier and bias b, the final $f(x)$ is

$$f\left(x\right) = \sum_{i=1}^{n} \alpha_i y_i K\left(x_i, x\right) + b \tag{3.9}$$

In case of regression, a penalty parameter value (C_i) for each sample of unbalanced data is used as given in Equation (3.10).

$$\underset{w,\xi,\xi^*}{min} \quad \frac{1}{2}\|w\|^2 + C_i \sum_{i=1}^{n} \left(\xi_i + \xi_i^*\right) \tag{3.10}$$

$$\begin{aligned} \text{s.t.} \quad & y_i - w \cdot \phi\left(x_i\right) - b \leq \varepsilon + \xi_i, \\ & w \cdot \phi\left(x_i\right) + b - y_i \leq \varepsilon + \xi_i^*, \\ & \xi_i, \xi_i^* \geq 0, \ \ i = 1, 2, \cdots, n. \end{aligned}$$

corresponding dual is

$$\max_{\alpha_i} \quad -\frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j K(x_i, x_j) - \varepsilon\sum_{i=1}^{n}|\alpha_i| + \sum_{i=1}^{n} y_i\alpha_i \tag{3.11}$$

$$\text{s.t.} \sum_{i=1}^{n} \alpha_i = 0, 0 \leq \alpha_i \leq C_i, i = 1, \cdots, n.$$

The final decisive function $f(x)$ is given by

$$f(x) = \sum_{i=1}^{n} \alpha_i K(x, x_i) + b \tag{3.12}$$

KKT conditions that are to be satisfied are given as

$|y_i - f(x_i)| \leq \varepsilon, \quad$ if $\quad \alpha_i = 0,$

$|y_i - f(x_i)| = \varepsilon, \quad$ if $\quad 0 < |\alpha_i| < C_i,$

$|y_i - f(x_i)| \geq \varepsilon, \quad$ if $\quad |\alpha_i| = C_i,$

$\sum_{i=1}^{n} \alpha_i = 0.$

3.3.2 Incremental SVM

Proposed approach for classification and prediction is trained over individual samples and learning is done incrementally by satisfying KKT conditions on all added samples to the training vector T [58]. The algorithm partitions training data T into three index sets as SupportSet(S), ErrorSet(E) and ReserveSet(R).

E represents Error vectors: $E = \{i : |\alpha_i| = C_i\}$

S represent Support vectors: $S = \{i : 0 < |\alpha_i| < C_i\}$

R represents Reserve vectors: $R = \{i : |\alpha_i| = 0\}$

During the learning phase, algorithm continuously checks for whether a new sample (x_c, y_c) added to the training data T can be inserted to reserve set R. If it is not possible to add new sample (x_c, y_c) in R, it tries to add this new sample in either support set S or error set E by maintaining equilibrium of the model on KKT conditions. Each update of the model migrates samples across index sets by passing them through the support set s, resulting in a change in the correlated **R** matrix accordingly. This method is repeated when all of the KKT constraints for the samples in T are met.

The margin function $g(x_i)$ for x_i is given by $g(x_i) = y_i - f(x_i)$. Let x_c be a new training point added to T, next initialize $\alpha_c = 0$ and progressively adjust the value of α_c to satisfy KKT criteria. The incremental relation between $\Delta g(x_i)$, $\Delta\alpha_i$ and Δb in accordance with [58] viz., :

$$\Delta g(x_i) = Q_{ic}\Delta\alpha_c + \sum_{j\in S} Q_{ij}\Delta\alpha_j + \Delta b, \qquad \forall i \in T \cup \{c\} \tag{3.13}$$

According to Equation (3.11),

$$\alpha_c + \sum_{i=1}^{n} \alpha_i = 0, \tag{3.14}$$

Integrating Equation (3.13), Equation (3.14) and KKT equations above, we obtain:

$$-Q_{ic}\Delta\alpha_c = \sum_{j\in S} Q_{ij}\Delta\alpha_j + \Delta b, \tag{3.15}$$

$$\sum_{j\in S} \Delta\alpha_j = -\Delta\alpha_c. \tag{3.16}$$

α_c is the coefficient being incremented. Let $S = \{s_1, s_2, \cdots, s_n\}$, thus Equation (3.16) can be represented in matrix form as:

$$\begin{bmatrix} 0 & 1 & \dots & 1 \\ 1 & Q_{s_1s_1} & \dots & Q_{s_1s_n} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & Q_{s_ns_1} & \dots & Q_{s_ns_n} \end{bmatrix} \begin{bmatrix} \Delta b \\ \Delta\alpha_{s_1} \\ \vdots \\ \Delta\alpha_{s_n} \end{bmatrix} = -\begin{bmatrix} 1 \\ Q_{s_1c} \\ \vdots \\ Q_{s_nc} \end{bmatrix} \Delta\alpha_c \quad ,$$

$$i.e. \begin{bmatrix} \Delta b \\ \Delta\alpha_{s_1} \\ \vdots \\ \Delta\alpha_{s_n} \end{bmatrix} = \beta \Delta\alpha_c \quad , \quad \beta = \begin{bmatrix} \beta \\ \beta_{s_1} \\ \vdots \\ \beta_{s_n} \end{bmatrix} = \textbf{-R} \begin{bmatrix} 1 \\ Q_{s_1c} \\ \vdots \\ Q_{s_nc} \end{bmatrix} \tag{3.17}$$

Defining a non-S set, $\check{S}$: $\check{S} = E \cup R = \{\check{s}_1, \check{s}_2, \cdots, \check{s}_n\}$. Combining Equation (3.15) and Equation (3.17), we get

$$\begin{bmatrix} \Delta g(x_{\check{s}_1}) \\ \Delta g(x_{\check{s}_2}) \\ \vdots \\ \Delta g(x_{\check{s}_n}) \end{bmatrix} = \gamma \Delta\alpha_c \quad ; \tag{3.18}$$

$$\gamma = \begin{bmatrix} Q_{\check{s}_1c} \\ Q_{\check{s}_2c} \\ \vdots \\ Q_{\check{s}_nc} \end{bmatrix} + \beta \begin{bmatrix} 1 & Q_{\check{s}_1s_1} & \dots & Q_{\check{s}_1s_n} \\ 1 & Q_{\check{s}_2s_1} & \dots & Q_{\check{s}_2s_n} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & Q_{\check{s}_ns_1} & \dots & Q_{\check{s}_ns_n} \end{bmatrix} \tag{3.19}$$

when $S = NULL$, Equation (3.16), Equation (3.17) and Equation (3.19) simplifies to: $\Delta g(x_{\check{s}}) = \Delta b, \forall \check{s} \in E \cup R$. The matrices above show the formula for moving samples from one index set to the next until the new data sample is correctly placed. An elaboration on how $\Delta\alpha_c$ value is generated and **R** matrix is updated is given in the following subsections.

3.3.2.1 Find suitable $\Delta\alpha_c$

Equation (3.17) and Equation (3.19) holds only when samples belonging to S remain unchanged. As a result, $\Delta\alpha_c$ is adjusted to a potentially huge number, either keeping S unchanged or terminating the algorithm. It is crucial to identify the direction in which the sample is moving before assessing all of the possible motions. The sample is anticipated to advance in the opposite direction of $g(x_c)$ until it reaches the margin $|\varepsilon|$: $sign(\Delta\alpha_c) = sign(-g(x_c))$ for learning purposes. To determine a bound on $\Delta\alpha_c$ imposed by each sample $\in T$, we consider different directions of the movement of samples $L_{c1}, L_{c2}, L_iS, L_iE$ and L_iR.

1. L_{c1}: Determines the distance between x_c and $|\varepsilon|$, then adds x_c to S and terminates

the algorithm.

2. L_{c2}: Determines the distance between x_c and C, x_c, then adds x_c to E and terminates the algorithm.

3. L_iS: Determines the distance between each $x_i \in S$ to E or to R. The direction of x_i is based on the value of $beta_i$ incurred by the direction of x_c. If the distance is negative, the sample will move in the opposite direction.

4. L_iE: Determines the distance between each $x_i \in E$ to S. The direction of x_i is based on the value of γ_i incurred by the direction of x_c. If the distance is negative, the sample will move in the opposite direction.

5. L_iR: determines the distance between each $x_i \in R$ to S.

The permitted $\Delta\alpha_c$ for each sample can be found using Equation (3.17) and Equation (3.18), with the final $\Delta\alpha_c$ being the smallest absolute value among all possible $\Delta\alpha_c$.

3.3.2.2 R matrix updation

R matrx used in Equation (3.17) is defined as :

$$\mathbf{R} = \begin{bmatrix} 0 & 1 & \dots & 1 \\ 1 & Q_{s_1 s_1} & \dots & Q_{s_1 s_n} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & Q_{s_n s_1} & \dots & Q_{s_n s_n} \end{bmatrix}^{-1} \tag{3.20}$$

By adopting the approach suggested by [76], the correlated matrix $\mathbf{R}$ can be updated efficiently. By initializing the **R** matrix as given in Equation (3.21), the first sample x_c can be readily inserted to S.

$$\mathbf{R} = \begin{bmatrix} 0 & 1 \\ 1 & Q_{cc} \end{bmatrix}^{-1} = \begin{bmatrix} -Q_{cc} & 1 \\ 1 & 0 \end{bmatrix} \tag{3.21}$$

When a new sample x_i is added to S, **R** can be updated as

$$\mathbf{R} = \begin{bmatrix} & & 0 \\ & \mathbf{R} & \vdots \\ 0 & \cdots & 0 \end{bmatrix} + \frac{1}{\gamma_i} \begin{bmatrix} \beta \\ 1 \end{bmatrix} \begin{bmatrix} \beta^T & 1 \end{bmatrix} \tag{3.22}$$

$$\beta = \text{-}\mathbf{R} \begin{bmatrix} 1 \\ Q_{s_1 i} \\ \vdots \\ Q_{s_n i} \end{bmatrix} ; \qquad \gamma_i = K_{ii} + \begin{bmatrix} 1 \\ Q_{s_1 i} \\ \vdots \\ Q_{s_n i} \end{bmatrix} \beta \tag{3.23}$$

When a sample $k \in S$ is removed from S, **R** can be obtained as: $\mathbf{R}_{ij} = \mathbf{R}_{ij} - \mathbf{R}_{kk}^{-1}\mathbf{R}_{ik}\mathbf{R}_{kj}$; $\forall j; i \neq k \in [0, \cdots, n]$

3.3.3 Modified Frank-Wolfe algorithm

Proposed modified FW algorithm generates a sequence of approximations $\{\alpha_k\}$ at each iteration k by computing optimal step-size analytically until it converges to an acceptable accuracy ε.

It does this using following steps:

(1) A linear approximation of $g(.)$ at current iteration α_k is performed to find the largest coordinate i^* and smallest coordinate j^* of the gradient $g(\alpha_k)$.

Let v_i denote the i^{th} vector of coordinate space. Then, v_{i^*} is the ascent vertex and v_{j^*} is the descent vertex.

Toward direction: $(v_{i^*} - \alpha_k)$; Away direction: $(\alpha_k - v_{j^*})$

(2) Each iteration α_k moves the current solution in the direction (Towards or Away) seeking for the best feasible improvement of the objective function.

(3) Use a search direction to update the next iteration (α_{k+1}) which is obtained by superimposing the above two steps.

$$\alpha_{k+1} = \frac{1}{2}(\alpha_k + \lambda(v_{i^*} - \alpha_k)) + \frac{1}{2}(\alpha_k + \lambda(\alpha_k - v_{j^*})) \tag{3.24}$$

Where λ is the step-size determined by line-search. The term $\lambda\alpha_k$ vanishes in the sum and only i^* and j^* components are updated.

$\alpha_{k+1} = \frac{1}{2}\alpha_k + \frac{1}{2}\lambda v_{i^*} - \frac{1}{2}\cancel{\lambda\alpha_k} + \frac{1}{2}\alpha_k + \frac{1}{2}\cancel{\lambda\alpha_k} - \frac{1}{2}\lambda v_{j^*}$

Simplifying the equation, the positive and negative term of $\lambda\alpha_k$ will be vanished.

$\alpha_{k+1} = \frac{1}{2}\alpha_k + \frac{1}{2}\lambda v_{i^*} + \frac{1}{2}\alpha_k - \frac{1}{2}\lambda v_{j^*}$

The sum of $\frac{1}{2}\alpha_k$ and $\frac{1}{2}\alpha_k$ is equal to α_k

$\alpha_{k+1} = \alpha_k + \frac{1}{2}\lambda v_{i^*} - \frac{1}{2}\lambda v_{j^*}$

By taking out the common term $\frac{1}{2}\lambda$, we get

$\alpha_{k+1} = \alpha_k + \frac{1}{2}\lambda(v_{i^*} - v_{j^*})$

Thus, the final update equation is:

$$\alpha_{k+1} = \alpha_k + \lambda(v_{i^*} - v_{j^*}) \tag{3.25}$$

This away step not only moves the solution away from the descent vertex v_{j^*}, but also moves closer to the ascent vertex v_{i^*} in the same iteration α_k. The proposed away step perturbs the current solution α_k only locally, results in change of weights of only v_{i^*} and v_{j^*} vertices by preserving weights all other active vertices same. Thus, away step does not increase the weights of the vectors corresponding to descent vertices v_{j^*}. These vectors may be spurious points that do not belong to the optimal solution and are eliminated from solution space.

Within the k iterations MFW solves the Equation (3.7). When the sample size k is increased, the updated approximation problem undergoes a number of additional steps. k can be increased in online learning as long as new samples are available. If the new training sample (x_c, y_c) can offer a better feasible direction than any of the previous samples in $S(k)$ in the following k^{th} iteration, it is chosen to be included in the new working set $S(k+1)$. Otherwise, for updating α and $g(\alpha)$, the best old sample within $S(k)$ will be chosen, and $S(k+1)$ will remain the same as $S(k)$. As a result, the working set's number of indices is $|S(k)| \leq k$, and $\alpha(k)$ has at most k nonzero items. The search direction $d^k = (v_{i^*} - v_{j^*})$ begins at the current solution $\alpha(k)$ and points to one of the unit simplex vertices. We can calculate the step-size analytically once the optimal vertex has been determined as: $\alpha_{k+1} = \alpha_k + \lambda(v_{i^*} - v_{j^*})$. As a result, the number of iterations required to achieve the necessary convergence is limited by $O(\frac{1}{\varepsilon})$, which is independent of the number of training samples.

A step by step procedure of above proposed method for classification and prediction estimation problems is given in Algorithm 5.1, which is a variant of classical SVM trained by incremental learning and optimized with modified FW method.

Algorithm 3.1 : Kernel SVM-MFW

```
Input: Dataset $X$, $\varepsilon$, $C$, $K(x_i, x)$, $\mathbf{R}$
Output: $\alpha_i, b, \mathbf{R}$
Initialization: $b = 0, S = 0, E = 0, R = 0, \alpha_c = 0$
 1: $f(x_c) \longleftarrow \alpha_c K(x_i, x) + b$
 2: $g(x_j) \longleftarrow f(x_j) - y_i$
 3: **if** $g(x_j) \leq |\varepsilon|$ **then**
 4:     $R \longleftarrow (x_j, y_j)$;
 5: **end if**
 6: **repeat**
 7:     $\Delta\beta = -R \cdot Q_{ic}$                    Q is a positive definite Kernel matrix
 8:     $\Delta\gamma = Q_{\check{s}c} + Q_{\check{s}i} \cdot \beta$              $\check{S}$ is a non $S$ set; $\check{S} \in E \cup R$; $\check{S} = \{\check{s}_1, \check{s}_2, \cdots, \check{s}_n\}$
 9:     $\Delta\alpha_c = min(L_{c1}, L_{c2}, L_iS, L_iE, L_iR)$
10:     **if** $\Delta\alpha_c == L_{c1}$ **then**
11:         $S \longleftarrow (x_c, y_c)$
12:         $\mathbf{R} = \mathbf{R} + \frac{1}{\gamma_i}\beta\beta^T$
13:     **else if** $\Delta\alpha_c == L_{c2}$ **then**
14:         $E \longleftarrow (x_c, y_c)$
15:     **else if** $\Delta\alpha_c == L_iS$ **then**
16:         **if** $\alpha_i == 0$ **then**
17:             $R \longleftarrow (x_i, y_i) \in S$
18:             $\mathbf{R}_{ij} = \mathbf{R}_{ij} - \mathbf{R}_{kk}^{-1}\mathbf{R}_{ik}\mathbf{R}_{kj}; k \in S$
19:         **else if** $\alpha_i == C$ **then**
20:             $E \longleftarrow (x_i, y_i) \in S$
21:             $\mathbf{R}_{ij} = \mathbf{R}_{ij} - \mathbf{R}_{kk}^{-1}\mathbf{R}_{ik}\mathbf{R}_{kj}; k \in S$
22:         **end if**
23:     **else if** $|g(x_i)| == \varepsilon$ **then**
24:         $S \longleftarrow (x_i, y_i) \in E$
25:         $\mathbf{R} = \mathbf{R} + \frac{1}{\gamma_i}\beta\beta^T$
26:     **else if** $|g(x_i)| == \varepsilon$ **then**
27:         $S \longleftarrow (x_i, y_i) \in R$
28:         $\mathbf{R} = \mathbf{R} + \frac{1}{\gamma_i}\beta\beta^T$
29:     **end if**
30: **until** $(x_c, y_c)$ satisfies KKT condition
```

3.4 Data used for experiments

Experiments were conducted on Parkinson's disease dataset which was taken from the PPMI repository (http://www.ppmi-info.org/data). dataset contains n =600 subjects of 195 healthy controls (HC) and 405 early PD subjects. A total of eleven discriminative features were used, out of which two predictors' description is given in Table 3.1 and Table 3.2 describes remaining nine predictors.

Table 3.1: Demographic data of study participants

Case(n=600)	Gender		Family History	
	Female	Male	Yes	No
Healthy Control (195)	65	130	10	185
Early PD (405)	141	264	98	307

Table 3.2: Mean and standard deviation of discriminative features of Age, MDS-UPDRS, MoCA, TD score, PIGD score, SBR values for left caudate(LtCd.SBR) and right caudate(RtCd.SBR), SBR values for left putamen(LtPt.SBR) and right putamen(RtPt.SBR).

Case($n = 600$)	*Age*	*MDS-UPDRS*	*MoCA*	*TD score*	*PIGD score*	*LtCd.SBR*	*RtCd.SBR*	*LtPt.SBR*	*RtPt.SBR*
HC	60.8±11.2	4.6±4.4	28.0±1.3	0±0	0±0	2.96± 0.6	2.93±0.5	2.12±0.5	2.12±0.5
Early PD	61.6±9.7	31.9±13.0	27.1±2.3	0.47±0.4	0.25±0.2	1.98±0.5	1.98±0.5	0.80±0.3	0.84±0.3

3.5 Experiments and Results

Data samples of n=600 with eleven predictors were used in the implementation of proposed SVM-MFW method. The dataset is divided into training and testing portions and is standardised to balance the influence of each feature. The model's hyper-parameters are optimized using a 10-fold grid search cross validation procedure. Leave-one-out cross validation (LOOCV) is used to assess the model's generalization performance. LOOCV takes a single sample of data for testing analysis and the rest is used for training. This procedure is carried out for each sample in the data set. Finally, the classification measures like accuracy, sensitivity and specificity, are estimated for the model. We also calculated a binary cross-entropy as a measure for predictive accuracy. Cross-entropy is a loss function used as measure for classification, whereas MSE and R^2 used for regression problems generally. As the proposed algorithm follows binomial distribution, it is good idea to use cross-entropy rather than MSE and R^2. Binary cross-entropy is given by E= $-y(log(p)+(1-y)log(1-p))$ where y is class label, $y \in (EarlyPD = -1, Healthy = +1)$ and p is the predicted probability of sample being early PD. For each sample being early PD (y=-1), it adds log(p(y)) to the loss (log(p(y)) is the log probability of sample being early PD. Conversely, for each sample being Healthy (y=+1), it adds log(1-p(y)) (log(1-p(y)) is the log probability of sample being Healthy. For implementation of proposed algorithm, we set optimal hyper parameter values as $\varepsilon = 2e^{-5}$, $C = 10$ and T_e(tolerance-error) = $1e^{-6}$. All runs of the proposed and existing models were performed on a computer with 3.40 GHz Intel i7 2600 CPU and 8 GB RAM using MATLAB 2017a.

The developed SVM-MFW is used with various kernels such as linear, polynomial, sigmoid, radial basis (RBF), and logistic functions to exhibit margin variations in classifying the data. We compared our work with the classical SVM over the same data with the above-mentioned kernels to validate the model's efficacy. The contour plots from Figure 3.3 to Figure 3.7 shows the comparison between the classical and proposed SVM-MFW. To explain further, the proposed algorithm minimizes the classification error by maximizing the ε-insensitivity zone and produces sparser representation of support vectors. The proposed model also provides class posterior probabilities for the early PD outcome and the contour divides the feature space into disjoint prediction regions which can assess the stage of pathology as shown in the Figure 3.3 to Figure 3.7. When compared to classical SVM, the SVM-MFW model has achieved significant margin distance (dotted lines) between two classes of data for different kernels, as well as a drastic reduction in the number of error vectors (black). At the same time, when compared to other kernels, the SVM-MFW with RBF kernel has shown a broad margin distribution with low error as given in Figure 3.3, Figure 3.4, Figure 3.6 and Figure 3.7. Despite the fact that the sigmoid kernel produces a huge margin distribution, the number of error vectors is much higher, as shown in Figure 3.6. As a result of the foregoing discussion, it can be concluded that the proposed models' performance in classifying the dataset is much superior and improves generalisation capabilities when compared to classical SVM.

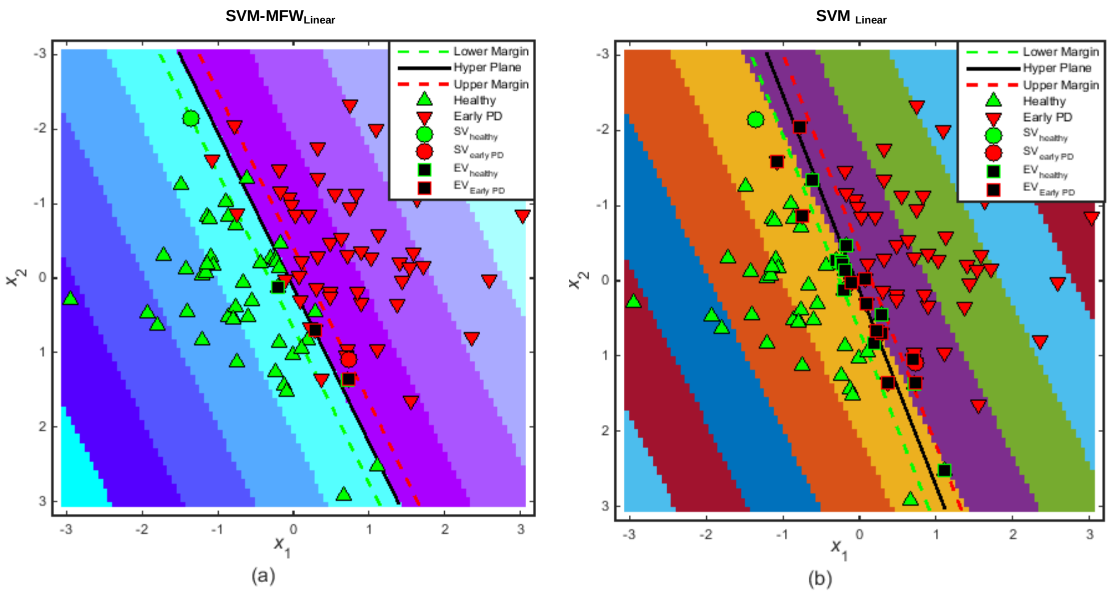


Figure 3.3: Contour plots showing the class separation of data using proposed SVM-MFW model in Figure(a) and classical SVM in Figure(b) along with predicted regions using Linear kernel. SV=support vectors and EV=error vectors.

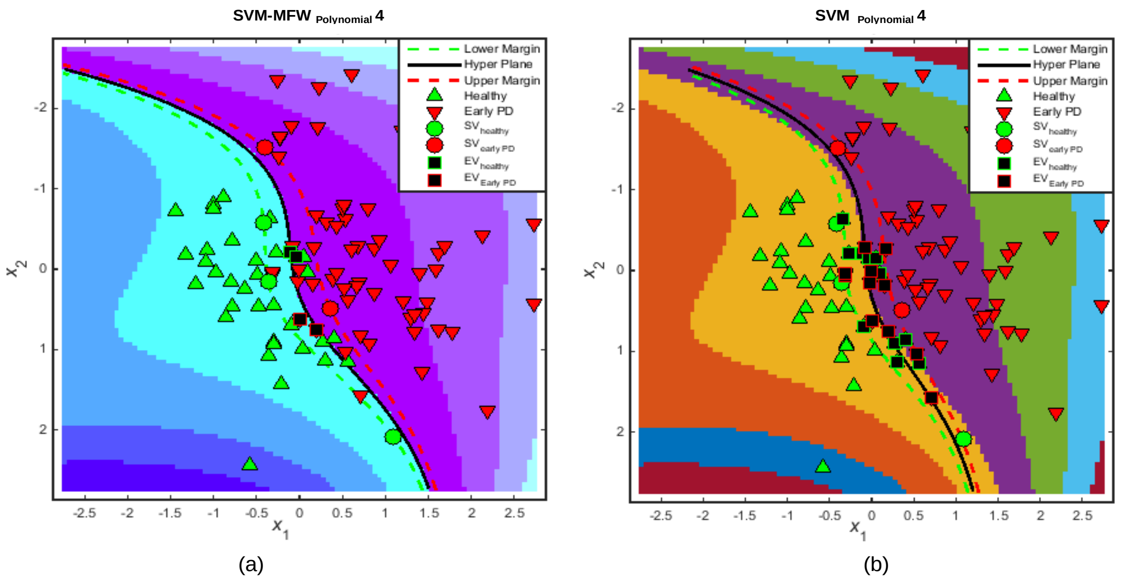


Figure 3.4: Contour plots showing the class separation of data using proposed SVM-MFW model in Figure(a) and classical SVM in Figure(b) along with predicted regions using polynomial kernel with degree 4. SV=support vectors and EV=error vectors.

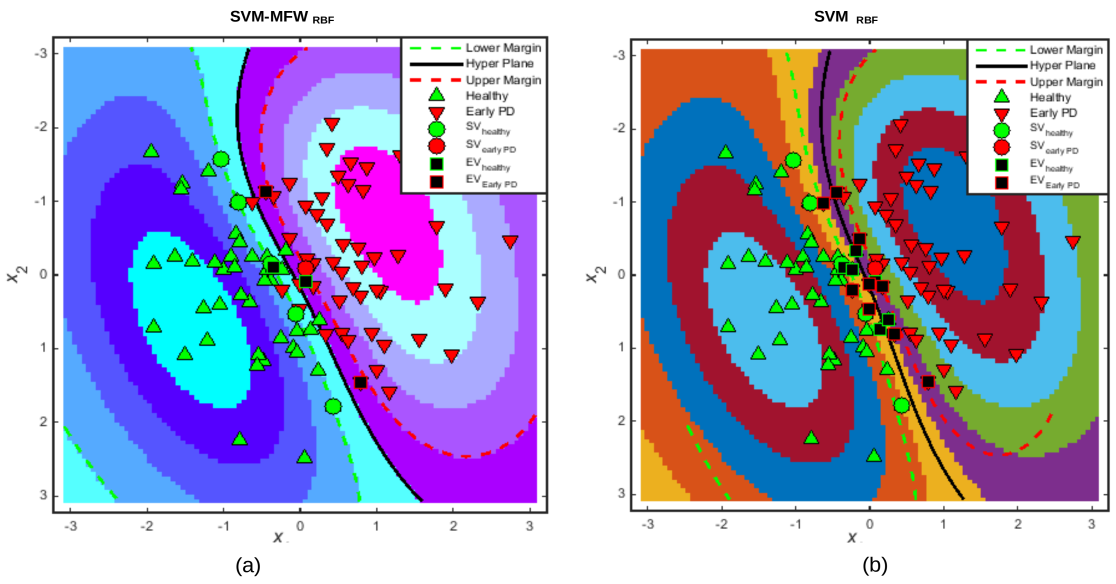


Figure 3.5: Contour plots showing the class separation of data using proposed SVM-MFW model in Figure(a) and classical SVM in Figure(b) along with predicted regions using RBF kernel. SV=support vectors and EV=error vectors.

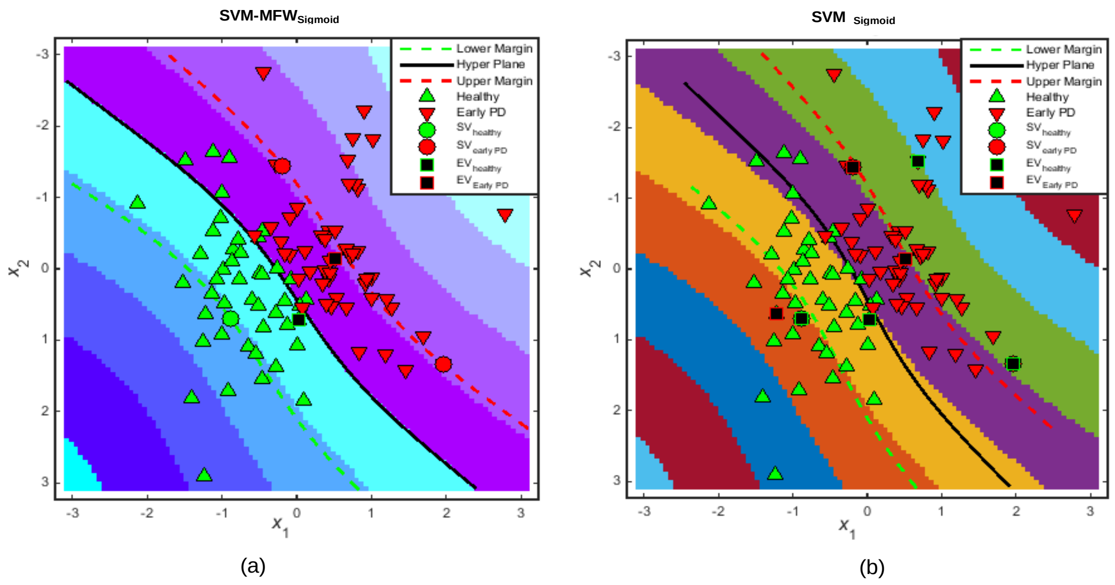
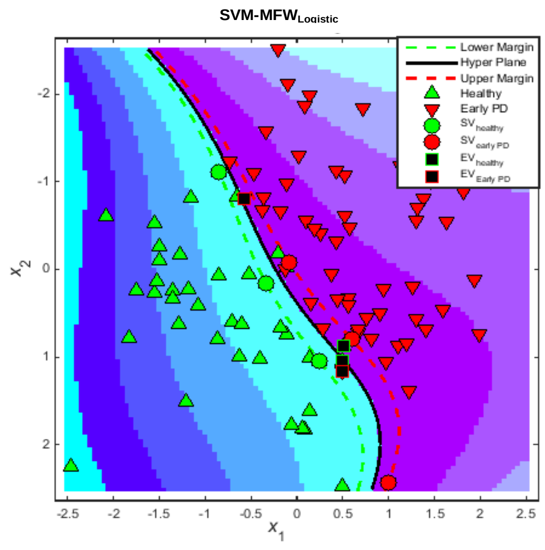
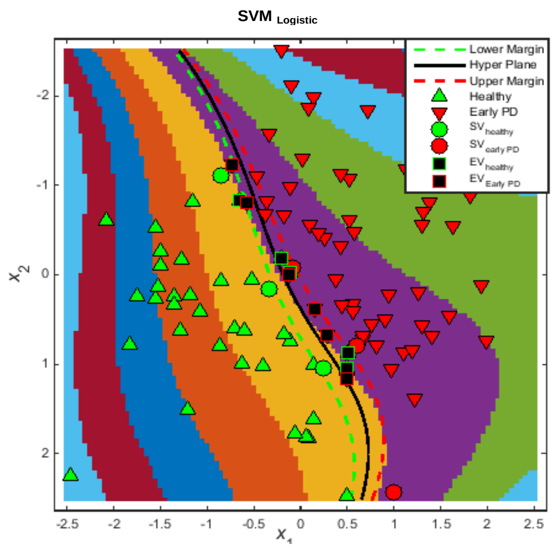


Figure 3.6: Contour plots showing the class separation of data using proposed SVM-MFW model in Figure(a) and classical SVM in Figure(b) along with predicted regions using sigmoid kernel. SV=support vectors and EV=error vectors.



(a)



(b)

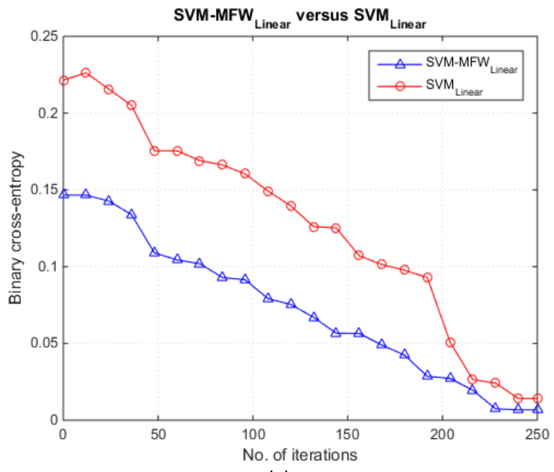
Figure 3.7: Contour plots showing the class separation of data using proposed SVM-MFW model in Figure(a) and classical SVM in Figure(b) along with predicted regions using logistic kernel. SV=support vectors and EV=error vectors.

The line plot of cross-entropy values versus number of iterations for different kernels to validate the prediction accuracy has shown in the Figure 3.8, which exhibits the line plot of cross-entropy values versus number of iterations for different kernels. When compared to classical SVM, the proposed technique achieves low cross-entropy values for each kernel. Thus, we may deduce from Figure 3.8 that our algorithm has a greater goodness of fit.

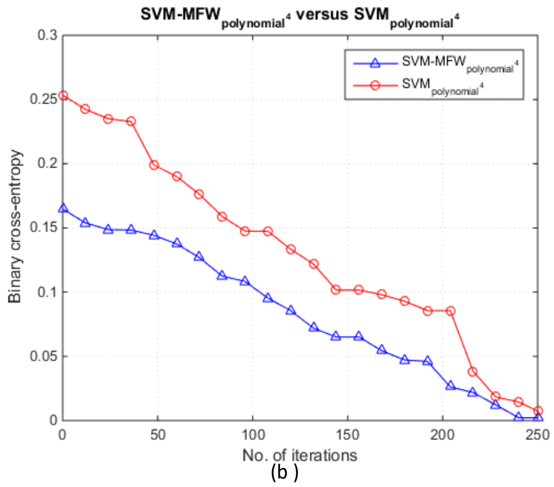
Table 3.3: Confusion matrix values and performance measures for the SVM-MFW and classical SVM with different kernels

Kernal	Model	$Accu(\%)$	$Sensi(\%)$	$Speci(\%)$	$CPUtime(s)$	$Cross-entropy$
$Linear$	SVM-MFW	98.00	98.89	96.01	1.78	0.106
	SVM	93.67	94.69	91.68	2.12	0.162
$Polynomial^4$	SVM-MFW	98.10	98.29	96.90	4.27	0.178
	SVM	92.70	94.20	90.40	5.61	0.231
Sigmoid	SVM-MFW	98.28	98.71	96.95	3.68	0.321
	SVM	94.70	95.02	93.92	3.92	0.420
RBF	SVM-MFW	**98.30**	98.51	97.90	2.32	0.134
	SVM	94.48	95.02	92.88	2.56	0.230
$Logistic$	SVM-MFW	98.10	98.52	97.33	3.43	0.183
	SVM	94.31	96.37	91.76	3.71	0.272

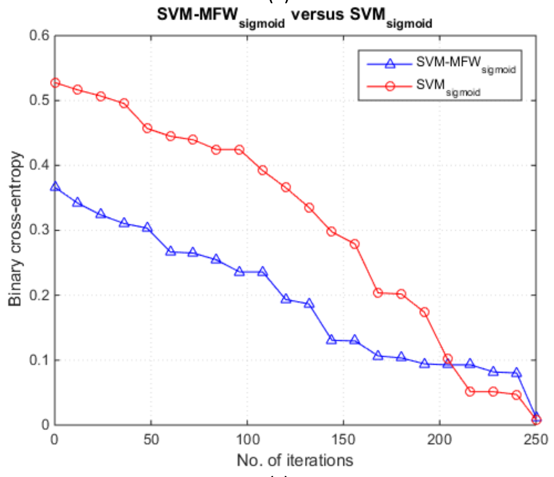
To show the effectiveness of the proposed algorithm in terms of computation time,



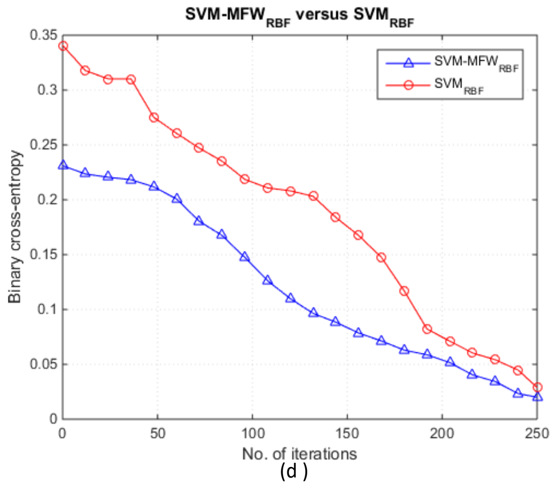
(a)



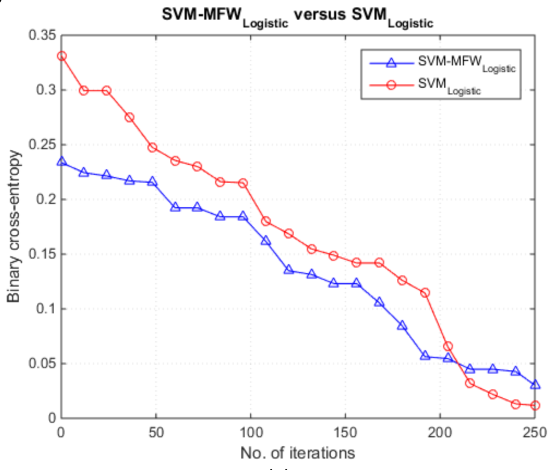
(b)



(c)



(d)



(e)

Figure 3.8: The binary cross-entropy of SVM-MFW and SVM algorithms with different kernels were compared. The line plot comparison is shown in Figure(a,b,c,d,e).

Table 3.4: Results of classification accuracy and CPU time for the SVM-MFW and existing classical ML techniques

Method	Accu (%)	Sensi (%)	Speci (%)	CPUtime(s)
MLR	94.03	96.01	91.52	2.13
NN	96.09	97.90	95.08	3.68
K-NN	95.81	96.95	92.68	3.71
SVM	94.56	95.32	92.81	2.57
proposed **SVM-MFW**	98.30	98.51	97.90	2.32

we perform experiments on same dataset using proposed model and classical SVM and provide the comparison of the two algorithms based on CPU time given in Figure 3.9. The proposed model with incremental learning and accelerated convergence using MFW technique significantly reduced the computation time when compared with non incremental classical SVM. From Figure 3.9, it is observed that SVM-MFW consumes less iteration time when compared to classical SVM in each of the kernels used.

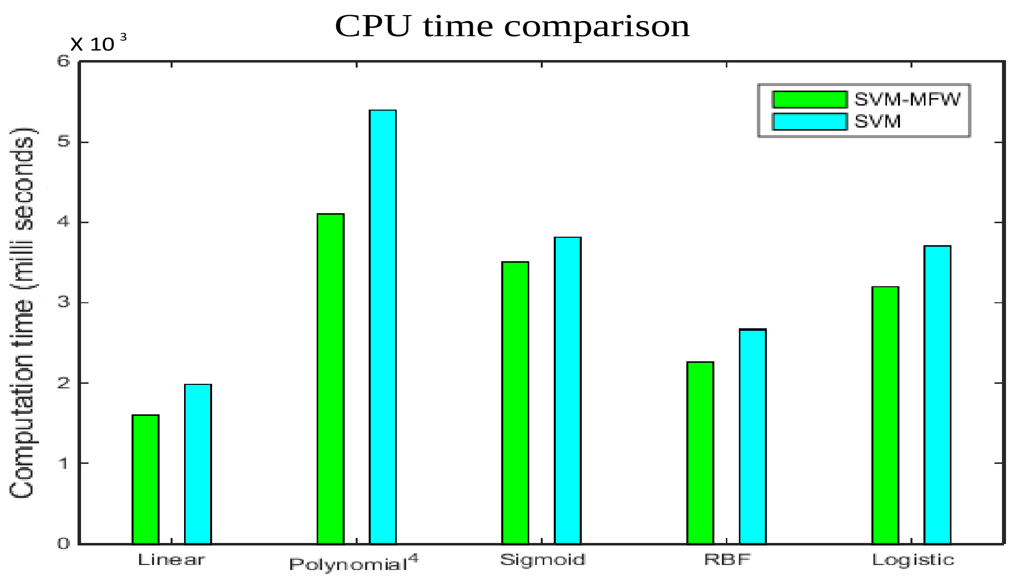


Figure 3.9: CPU time comparison of SVM-MFW and classical SVM with various kernels used in the model building

The confusion matrix values and performance measures for the proposed SVM-MFW and classical SVM algorithms with linear, polynomial of order 4, sigmoid, RBF, and Logistic kernels are summarised in Table 3.3. According to the Table 3.3, SVM-MFW has a high classification accuracy of 98.3%, compared to 94.4% for classical SVM using RBF kernel. The validation of proposed algorithm with RBF kernel is shown in Table 5.2. The model correctly classified 191 healthy controls out of 195 and 399 early PD patients out of 405, while it misclassified 4 healthy controls as early PD group and 6 early PD group as healthy controls. The optimal classification accuracy of 98.3% is achieved using RBF kernel. From above analysis, we can find that the proposed model with selected discriminative features has improved the performance of PD diagnosis in terms of accuracy and computation time.

Table 3.5: Confusion matrix of SVM-MFW with RBF kernel

Actual group	Predicted group		
	Healthy	PD	% Accuracy
Healthy	191	4	97.9
Early PD	6	399	98.5
Overall %			98.3

The present work is also compared with other classical ML algorithms like basic NN, K-NN and LR in terms of classification accuracy and computation time. We apply these techniques on the same dataset and results are presented in Table 3.4. We applied Wilcoxon Signed-Rank Test [77] to assess the statistical significance of the accuracies obtained by these algorithms against accuracy of the proposed work. The obtained accuracies are similar to some extent but are statistically significant with $\rho < 0.05$. It can be observed from Table 3.4 that the proposed algorithm gives better results in comparison with the classical ML techniques.

3.6 Summary

In the present work, we proposed a new technique called SVM-MFW using incremental version of SVM incorporated with modified Frank-Wolfe method. The proposed model used the away step methodology to speed up the training process with less number of iterations and converges faster to an acceptable accuracy, thereby reducing the overall computation time of algorithm. The present work is also validated on classification accuracies and computational time against other state-of-the-art techniques. It is shown that the

SVM-MFW algorithm comparatively achieves better performance (98.3% accuracy) in less amount of time than its counterparts. Therefore, we state that our model can solve classification and prediction problems in less computational time over incremental datasets.

Chapter 4

ε-Support vector regression optimized with Large Margin Distribution using modified dual coordinate descent strategy ε-MDSVR)

This chapter discusses the proposed ε-Support vector regression algorithm that is optimized with the Large margin distribution (LDM) machine by employing modified dual coordinate descent (DCD) technique to enhance the performance. The chapter introduces the support vector regression (SVR) in brief and describes the concept of optimization of margin distribution to address the issue of outliers. Margin distribution is implemented by modified DCD technique to increase the learning speed of algorithm. Experimental results on four benchmark UCI datasets revealed the effectiveness of the proposed model against the classical SVR.
The main contributions of this chapter are described below:

- The proposed ε-MDSVR model attempts to make full use of the training set to avoid over-fitting while also minimizing scattering of the data in ε-tube simultaneously.
- By improving the margin distribution, the model obtains higher generalization performance..

- This margin distribution is characterized by the mean and variance.
- Proposed algorithm maximizes the mean by considering the distance between the data points nearer to the margin and later attempts to minimize the margin variance. This can be done by employing Dual coordinate descent method (DCD).
- The modified DCD method orderly updates one variable at a time. The variable is selected for updation such that if it possibly derives the maximum optimization in objective function value. This strategy can increase the learning speed and improve the generalization performance.
- The model is experimented on four popular UCI datasets and validated in terms of Mean square error (MSE), coefficient of regression (R^2) and CPU time using Linear and RBF kernels.

4.1 Introduction

SVM is popular in classification problems and can be used for regression estimation as well. The soft margin SVM enabled its use in estimating the relationships between data values, this technique is known as ε-SVR. Maximum margin is the fundamental issue of SVMs. SVM considers a single point margin optimization. The function $f(x)$ will not change by adding a new sample (x_i, y_i) as long as $f(x_i)$ does not deviate more than ε margin from y_i, moreover, deviations are penalized. ε-SVR are harmed by the amount of data in the hyper plane, they are weak for non-significant deviants. If the data distribution within the ε-tube differs significantly from the direction of the support vectors (outliers), the final fitting function may be inconsistent with the real data distribution and ineffective. ε-SVR is not very robust to the outliers, For determining the predictive curve, SVR model completely disregards all data points that fall within the ε-tube. This method sparsifies the SVR model but does not reduce scatter inside the ε-tube. Aside from sparsity, we also need to reduce the scatter of data points within the ε-tube.

According to recent theoretical findings, the margin distribution, rather than a single point margin, is a viable technique to address outliers and hence improve SVR generalisation performance. Thus, the Large margin distribution (LDM) is proved to be essential to boost the generalization performance of SVR. [37, 38]. By concurrently increasing the margin mean and lowering the margin variance, the margin distribution can be optimized. The learning speed of Margin distribution can be increased by employing the dual coordinate descent (DCD) technique for kernel SVM [37]. The DCD algorithm can solve the QP of SVM by sequentially optimizing one variable in sub-problems of SVM. It primarily comprises of outer and inner iterations, with each outer iteration including n samples of inner iteration. At each inner iteration, the DCD algorithm updates all SVM variables in a systematic manner. By tackling the sub-problems in a random order at each outer iteration, the DCD can enhance the SVM learning speed. However, because this fundamental random update technique of variables one by one frequently fails to achieve an effective decrease in the SVM's objective value, learning pace is slowed.

To address the issue of blindfolded update of variables in the basic DCD, we proposed a modified DCD algorithm to increase the learning speed of SVR further. The modified DCD technique updates one variable in a sub-problem in an orderly manner, with this variable being chosen if it is likely to result in the greatest decrease in the objective value. That is, for each iteration, a variable is chosen that may result in the greatest drop in the objective value, thereby getting the solution closer to its optimum. In comparison to the standard DCD algorithm, this modified DCD technique not only effectively addresses the issue of tedious update in iterations, but also provides a simpler formulation. This proposed ε-MDSVR algorithm technique can concurrently boost learning speed and generalization performance. Experimental results show that the proposed ε-MDSVR approach achieves greater fitting quality and numerical convergence than the classical ε-SVR algorithm. Therefore, the proposed work can have a trade-off between computational complexity and generalization performance.

The rest of the content of this chapter is organized as follows. Section 4.2 presents the basic equations used to derive the proposed model. Implementation details of the proposed algorithm are described in Section 4.3. Experiment results and validation of the proposed model and its behaviour against other recent existing research is discussed in Section 4.4. Chapter is summarised in section 4.5.

4.2 Preliminaries

This section gives the basic equations of Large margin distribution machines and its optimization. We also describe the dual coordinate descent strategy.

4.2.1 Large margin distribution SVM

SVM works on the principle of increasing the minimum width of the margin i.e., minimum distance from the training points to the separating hyperplane. SVM tries to minimize the residual of the i^{th} data point. The LDM model proposed by [37, 38] revealed that margin distribution, rather than a single-point margin, is more important for SVM generalization performance. The goal function of SVM may be tuned for margin distribution by increasing the margin mean and decreasing the margin variance. Let X be the matrix with column vectors $X = [\phi(x_i), \cdots, \phi(x_n)]$ and let $y = [y_i, \cdots, y_n]^T$, then the objective function of SVM is $f(x) = w.\phi(x)$. The geometric margin is the minimum width between instances to the separating hyperplane. For classification it is given as: $\frac{y_i(w\cdot\phi(x_i)+b)}{\|w\|}$. For Regression it is given as: $\frac{|w\cdot\phi(x_i)-y_i|}{\|w\|}$. The functional margin for the i^{th} data point in classification is: $u_i = y_i(w \cdot \phi(x_i) + b)$; In regression, $\gamma = (w.\phi(x_i) - y_i)$;
In SVM, the margin of a training instance (x_i, y_i) is given by

$$\gamma_i = y_i w^T \phi(x_i), \qquad i = 1, \cdots, n. \tag{4.1}$$

According to Equation (4.1), the margin mean is

$$\bar{\gamma} = \frac{1}{n}\sum_{i=1}^{n} y_i w^T \phi(x_i) = \frac{1}{n}(Xy)^T w, \tag{4.2}$$

and the margin variance is

$$\tilde{\gamma} = \frac{1}{n^2} \sum_{i,j=1}^{n} (y_i w^T \phi(x_i) - y_j w^T \phi(x_j))^2 = \frac{2}{n^2}(nw^T X X^T w - w^T X y y^T X^T w) \tag{4.3}$$

The linear LDM model discoveres the separating hyperplane $f(x) = 0$ by solving the following optimization problem:

$$\begin{aligned} \min_{w,\xi} \quad & \frac{1}{2} w^{\mathrm{T}} w + \lambda_1 \bar{\gamma} - \lambda_2 \tilde{\gamma} + C \sum_{i=1}^{n} \xi_i \\ & \text{subject to,} \\ & y_i \left(w^{\mathrm{T}} x_i \right) \geq 1 - \xi_i, \xi_i \geq 0, i = 1, 2, \ldots, n. \end{aligned} \tag{4.4}$$

The parameter λ_i is used to achieve a trade-off between the mean of functional margin and model complexity. The positive parameters λ_1 and λ_2 are used to trade off the margin mean and variance. When λ_1 and λ_2 are equal to zero, the LDM formulation becomes SVM.

4.2.2 Dual coordinate descent technique

The dual coordinate descent (DCD) technique resolves the dual issue of SVR by minimising chunks of sub-problems in a random order of inner iterations. At each iteration l, the DCD algorithm starts with an initial approximation, let $\alpha^{(0)} \in \mathcal{R}^n$, and generates a succession of approximations $\left\{ \boldsymbol{\alpha}^{(l)}, l \geqslant 0 \right\}$. It consists of outer and inner iterations, and updates $\alpha^{(l)}$ to $\boldsymbol{\alpha}^{(l+1)}$ at each outer iteration which simultaneously updates $\alpha_1, \ldots, \alpha_n$ at each inner iterations of n. Thus, the following vectors are generated at each outer iteration.

$\boldsymbol{\alpha}^{(l),i} \in \mathcal{R}^n, i = 1, \ldots, n+1, \quad$ s.t

$\boldsymbol{\alpha}^{(l),1} = \boldsymbol{\alpha^l}, \quad \boldsymbol{\alpha}^{(l),n+1} = \boldsymbol{\alpha}^{(l+1)},$

$\boldsymbol{\alpha}^{(l)i} = \left[\alpha_1^{(l+1)}; \ldots; \alpha_{i-1}^{(l+1)}; \alpha_i^{(l)}, \ldots; \alpha_n^{(l)} \right], i = 1, \cdots, n.$

For updating $\alpha^{(l),1}$ to $\alpha^{(l),i+1}$, the DCD algorithm solves the following one variable sub-problem as:

$$\min_{\lambda} f\left(\alpha^{(l),i} + \lambda \varepsilon_i \right) \quad s.t. \quad 0 \leqslant \alpha_i^{(l)} + \lambda \leqslant C, \quad \varepsilon_i = [0; \ldots; 1; \ldots; 0] \tag{4.5}$$

The objective function of Equation (4.5) is a simple quadratic function of λ :

$$f\left(\alpha^{(l),i}+\lambda\varepsilon_{\mathrm{i}}\right)=\frac{1}{2}Q_{\mathrm{ii}}\lambda^{2}+\nabla_{i}f\left(\boldsymbol{\alpha}^{(l),i}\right)\lambda+constant \tag{4.6}$$

where $\nabla_i f$ is the i^{th} component of the gradient ∇f, which is defined as: $\nabla_i f(\boldsymbol{\alpha})=(\boldsymbol{Q\alpha})_i-e_i=\boldsymbol{\alpha}^{\top}Q_i-e_i=\sum_{j=1}^{n}Q_{ij}\alpha_j-e_i$.
Q_i is the i^{th} column of Q. It can be seen that Equation (4.5) has an optimum at $\lambda=0$, i.e., α_i is not required to update if $\nabla_i^p f\left(\boldsymbol{\alpha}^{(l),i}\right)=0$, where $\nabla_i^p f(\boldsymbol{\alpha})$ is the projected gradient. The same can be represented in following Equation (4.7)

$$\nabla_i^p f(\boldsymbol{\alpha})=\begin{cases}\min\left(\nabla_i f(\boldsymbol{\alpha}),0\right), & \text{if } \alpha_i=0\\ \max\left(\nabla_i f(\boldsymbol{\alpha}),0\right), & \text{if } \alpha_i=C\\ \nabla_i f(\boldsymbol{\alpha}), & \text{otherwise}\end{cases} \tag{4.7}$$

If $\nabla_1^p f\left(\boldsymbol{\alpha}^{(l),i}\right)=0$ holds, we move to the next index of $(i+1)$ without updating $\boldsymbol{\alpha}^{(l)}$. Otherwise, the solution of Equation (4.5) can be given in Equation (4.8).

$$\alpha_i^{(l),i+1}=\min\left(\max\left(\alpha_i^{(l),i}-\frac{\nabla_i f\left(\boldsymbol{\alpha}^{(l),i}\right)}{Q_{ii}},0\right),C\right) \tag{4.8}$$

4.3 Proposed ε-MDSVR

The ε-SVR tries to reduce the ε-insensitive loss function, which overlooks error points up to ε. Support vectors are locations on the fitting boundary curves of $f(x)+\varepsilon$, $f(x)-\varepsilon$, and outside the ε-insensitive zone that will be used to create the final regressor. To achieve sparsity, sample points inside the boundary curve were omitted, however this also leads ε-SVR to lose the information contained in the training set. Outliers that are meant to be beyond the ε-insensitive zone have an effect on the regressor curve's orientation and position, leading to over-fitting. As a result, the ε-SVR model performs poorly on noisy datasets. Rather than only maximizing the minimal margin, the proposed ε-MDSVR model optimizes the margin mean while minimising the margin variance. In the building of the regressor curve, the suggested model assigns certain weights to the points that are inside the ε-insensitive zone. As a result of this method, the proposed model is insensitive to noisy data samples (outliers) and so avoids the problem of over-fitting.

4.3.1 Large margin distribution SVR

ε-SVR formulation:

$$\begin{aligned} &\min_{w,\xi,\xi^*} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i+\xi_i^*) \\ &\text{s.t. } y_i - w\cdot\phi(x_i) - b \leq \varepsilon + \xi_i, \\ &w\cdot\phi(x_i) + b - y_i \leq \varepsilon + \xi_i^*, \quad \xi_i,\xi_i^* \geq 0, i=1,2,\ldots,n \end{aligned} \tag{4.9}$$

To measure the empirical risk, it uses the ε-insensitive loss function.
$R_{\text{emp}}^{\varepsilon} = \frac{1}{n}\sum_{i=1}^{n}|y_i - f(x_i)|_{\varepsilon}$, where $|y_i - f(x_i)|_{\varepsilon} = \max(0, |y_i - f(x_i)| - \varepsilon)$.
ε-SVR model minimizes the ε-insensitive loss function with a regularization term $\frac{1}{2}\|w\|^2$ in its optimization problem.

The functional margin in Support vector regression can be described as a difference between the real values and estimated values of objective function. The mean of functional margin in regression is: $\check{\gamma} = \frac{1}{n}(w.\phi(x_i) - y_i)^2$

$$\check{\gamma} = \frac{1}{n}\sum_{i=1}^{n}(w.\phi(x_i) - y_i)^2 = \frac{1}{n}\left(w^T\phi(X)\phi(X)^Tw - 2(\phi(X)Y)^Tw + YY^T\right) \tag{4.10}$$

where $\phi(X) = [\phi(x_1),\ldots,\phi(x_n)]$ and $\phi(X)\phi(X)^T = \sum_{i=1}^{n}\phi(x_i)\phi(x_i)$

Let $f(x) = w^{\mathrm{T}}x + b$, where $w \in R^n$, the proposed LDM minimizes the following generalized loss function of SVR along with the regularization term, $k > 0$ and $C > 0$.

$$min \quad R_{\text{emp}}^{f} = \frac{k}{2}\sum_{i=1}^{n}(y_i - f(x_i))^2 + C\cdot\sum_{i=1}^{n}|y_i - f(x_i)|_{\epsilon} \tag{4.11}$$

By introducing the regularization term $\frac{1}{2}\|w\|^2$ and slack variables ξ_i and ξ_i^*, the primal form of above equation is:

$$\min_{wb,\xi_1,\xi_2} \frac{c}{2}\|w\|^2 + \frac{k}{2}\sum_{i=1}^{n}(y_i - (w \cdot \phi(x_i) + b))^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \tag{4.12}$$

s.t. $\quad y_i - (w \cdot \phi(x_i) + b) \leq \epsilon + \xi_i; \quad (w \cdot \phi(x_i) + b) - y_i \leq \epsilon + \xi_i^*; \quad \xi_i, \xi_i^* \geq 0.$

where C, k, ϵ and c are user-defined parameters with positive values. The proposed ε-MDSVR model minimizes three terms in its optimization problem:

1. $\frac{1}{2}w^{\mathrm{T}}w$ makes the regressor as flat as possible.
2. $\sum_{i=1}^{l}(y_i - f(x_i))^2$ minimizes the scattering of the data points.
3. $\sum_{i=1}^{l}|y_i - f\,(x_i)\,|$ produces required sparsity.

These three elements in the objective function are suitably traded off to make full use of the training set while avoiding model over-fitting at the same time.

When $k = 0$, the primal problem of proposed ε-MDSVR model in Equation (4.12) simplifies to primal problem of ε-SVR as in Equation (4.9). Also, when C becomes zero in the primal problem of the proposed ε-MDSVR formulation in Equation (4.12), the variables $\xi_i, \xi_i^* \geq 0$ are no more minimized in Equation (4.12) and hence can take any values. As a result, the optimization problem's constraints are meaningless because they are always satisfied for any choice of (w, b). So with $C = 0$, the proposed model only minimizes $\frac{c}{2}\|w\|^2 + \frac{k}{2}\sum_{i=1}^{n}(y_i - (w \cdot \phi(x_i) + b))^2$ in its optimization problem. By deriving the equivalent dual problem, the answer to the primal problem in Equation (4.12) can be discovered. The primal form of Equation (4.12) can be turned into a dual formulation with Lagrange multipliers using the suggestions of [16] and the optimal solution of w in [37] as follows:

$$\begin{aligned} L(\alpha, \alpha^*, \beta, \beta^*) &= \frac{c}{2}vI_0v + \frac{k}{2}(Y - Hv)^{\mathrm{T}}(Y - Hv) \\ &+ C\sum_{i=1}^{n}(\xi_i + \xi_i^*) - \sum_{i=1}^{n}(\beta_i\xi_i + \beta_i^*\xi_i^*) \\ &- \sum_{i=1}^{n}\alpha_i(\epsilon + \xi_i - y_i + Hv) - \sum_{i=1}^{n}\alpha_i^*(\epsilon + \xi_i^* + y_i - Hv) \end{aligned} \tag{4.13}$$

where $H = \phi(X)^T\phi(X)$ and $v = \begin{bmatrix} w \\ b \end{bmatrix}$, then $\|w\|^2$ can be written as $\|w\|^2 = v^{\mathrm{T}}I_0v$, where $I_0 = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}$ and I is $n \times n$ identity matrix.

The KKT optimality conditions are given as follows:

$$\begin{aligned}
&\frac{\partial L}{\partial v}=\left(cI_0+kH^{\mathrm{T}}H\right)v-H^{\mathrm{T}}Y-H^{\mathrm{T}}\alpha_i+H^{\mathrm{T}}\alpha_i^*=0\\
&\frac{\partial L}{\partial \xi_i}=C-\alpha_i-\beta_i=0\\
&\frac{\partial L}{\partial \xi_i^*}=C-\alpha_i^*-\beta_{i^*}=0\\
&Y-Hv\le\epsilon+\xi_i,\quad \xi_i\ge 0\\
&Hv-Y\le\epsilon+\xi_i^*,\quad \xi_i^*\ge 0\\
&\alpha_i(\epsilon+\xi_i-y_i+Hv)=0\\
&\alpha_i^*(\epsilon+\xi_i^*+y_i-Hv)=0,\\
&\beta_i\xi_i=0,\quad \beta_i^*\xi_i^*=0\\
&\alpha_i\ge 0,\quad \alpha_i^*\ge 0,\quad \beta_i\ge 0,\quad \beta_i^*\ge 0.
\end{aligned}\tag{4.14}$$

Using the above KKT conditions, Equation (4.12) can be obtained as

$$\begin{aligned}
&\min_{\alpha_i,\alpha_i^*}\frac{1}{2}\left(\alpha_i-\alpha_i^*\right)H\left(cI_0+kH^{\mathrm{T}}H\right)^{-1}H^{\mathrm{T}}\left(\alpha_i-\alpha_i^*\right)\\
&\quad+Y^{\mathrm{T}}H\left(cI_0+kH^{\mathrm{T}}H\right)^{-1}H^{\mathrm{T}}\left(\alpha_i-\alpha_i^*\right)\\
&\quad-Y^{\mathrm{T}}\left(\alpha_i-\alpha_i^*\right)+\epsilon\left(\alpha_i+\alpha_i^*\right)\\
&\text{subject to,}\\
&0\le\alpha_i\le C\\
&0\le\alpha_i^*\le C.
\end{aligned}\tag{4.15}$$

After obtaining the optimal value of the α and α^* from Equation (4.15), we can obtain v using Equation (4.14) as:

$$v=\begin{bmatrix} w\\ b\end{bmatrix}=\left(cI_0+kH^{\mathrm{T}}H\right)^{-1}H^{\mathrm{T}}\left(\alpha-\alpha^*+Y\right).$$

Thus, for the given $x\in R^n$, the estimated regressor is obtained as in Equation (4.16)

$$f(x)=\sum_{i=1}^{n}\left(\alpha_i-\alpha_i^*\right)K\left(x_i,x\right)\tag{4.16}$$

By establishing a fair trade-off between the insensitive loss function and the quadratic loss function via the user-defined parameters c, k, and C, the Proposed ε-MDSVR model achieves higher generalisation ability. To solve Equation (4.15), we use a modified DCD approach. As explained in the following section, the proposed DCD technique constantly selects one variable for reduction and keeps the others as constants at each iteration.

4.3.2 Modified DCD

The modified DCD method orderly updates one variable at each iteration from a set of sub-problems. The variable is chosen if it has the greatest potential for lowering the objective value. That is, for each iteration, a variable is chosen that may result in the greatest drop in the objective value, thereby getting the solution closer to its optimum. This strategy not only effectively overcomes the issue of possible spurious update in the iterations but also gives simpler formulation compared to the basic DCD algorithm. This proposed ε-MDSVR algorithm strategy can increase the learning speed and improve the generalization performance.

Let the dual form of SVR can be represented as:

$$\min_{\alpha} f(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^\top \boldsymbol{Q}\boldsymbol{\alpha} - \boldsymbol{e}^T\boldsymbol{\alpha} \quad \text{s.t.} \quad 0 \leqslant \alpha_i \leqslant C, \quad i = 1, \ldots, n. \tag{4.17}$$

where $\boldsymbol{Q}$ is a matrix with $Q_{ij} = y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j$, and $\boldsymbol{e}$ is a n dimensional vector. $\boldsymbol{e} = [1; \ldots; 1]$.

Let the initial solution be represented as:
$f(0) = \frac{1}{2}\ \boldsymbol{\alpha}^T\boldsymbol{Q}\boldsymbol{\alpha} - \boldsymbol{e}^T\boldsymbol{\alpha}$. The improved DCD analyses only one component of α that needs to be updated at each iteration, rather than any outer or inner iteration. The index $\alpha_L \rightarrow \alpha_L + \lambda, L \in \{1, \ldots, n\}$ is to be updated at each iteration.

$$\begin{aligned} f(\lambda) &= \frac{1}{2}\left(\alpha_L + \lambda\right)^2 Q_{LL} + \frac{1}{2}\alpha_N^T Q_{NN}\alpha_N + \left(\alpha_L + \lambda\right)\alpha_N^\top Q_{NL} - e_L\left(\alpha_L + \lambda\right) - e_N^T\alpha_N \\ &= f(0) + \frac{1}{2}\lambda^2 Q_{LL} - \lambda\left(e_L - \alpha_N^\top Q_{NL} - Q_{LL}\alpha_L\right) = f(0) + \frac{1}{2}\lambda^2 Q_{LL} - \lambda\left(e_L - \alpha^\top Q_{.,L}\right) \end{aligned} \tag{4.18}$$

where $\mathcal{N}$ is the index set $\{1, \ldots, n\}\backslash\{L\}$ and $Q_{.,L}$ is the L^{th} column of Q.
The derivation of λ can be given as:

$$\frac{df(\lambda)}{d\lambda} = 0 \Rightarrow \lambda = \frac{e_L - \alpha^T\mathbf{Q}_{.,L}}{Q_{LL}}, \tag{4.19}$$

$$f(\lambda) = f(0) - \frac{\left(e_L - \alpha^T Q_{.,L}\right)^2}{2Q_{LL}} \tag{4.20}$$

The objective decrease will now be approximately largest if we maximize $\left(e_L - \alpha^T Q_{.,L}\right)^2 / Q_{LL}$, which chooses the L index as:

$$L = \arg \max_{1 \leq i \leq n} \left\{ \frac{\left(e_i - \boldsymbol{\alpha}^T \boldsymbol{Q}_{.,i}\right)^2}{Q_{ii}} \right\}. \tag{4.21}$$

Therefore, we have a simple update $\alpha_L^{\text{new}} = \alpha_L + \lambda$ with maximum possible decrease in objective function. To reduce the λ value such that $0 \leqslant \alpha_L^{\text{new}} \leqslant C$, we adjust the L index as:

$$L = \arg \max_{i \in A} \left\{ \frac{\left(e_i - \boldsymbol{\alpha}^T \boldsymbol{Q}_{.,i}\right)^2}{Q_{ii}} \right\}, \tag{4.22}$$

where the index set A is:

$$\mathcal{A} = \left\{ i : \alpha_i > 0 \text{ if } \frac{e_i - \boldsymbol{\alpha}^T \boldsymbol{Q}_{.,i}}{Q_{ii}} < 0 \quad \text{or} \quad \alpha_i < C \text{ if } \frac{e_i - \boldsymbol{\alpha}^T \boldsymbol{Q}_{.,i}}{Q_{ii}} > 0 \right\} \tag{4.23}$$

Finally, the above formulations demonstrate that there exists not only maximization in minimum margin but also an optimization in the margin distribution concurrently to obtain a superior trade-off between the distribution of the entire training data and the distribution of support vectors. To validate the performance of this algorithm we used popular matrices of MSE and R^2. Experimental results on benchmark datasets show that the proposed ε-MDSVR algorithm obtains better prediction accuracy with faster numerical convergence than the ε-SVR algorithm. Therefore, the proposed work can have a trade-off between sparsity and scatter minimization.

4.4 Experiments and Results

This section gives the detailed experimental study of the proposed method to show the effectiveness in terms of popular regression metrics and computation time over four benchmark datasets. The results are compared with other standard regression techniques to assess the fit quality.

4.4.1 Experimental setup

To evaluate the model, experiments are performed on four benchmark datasets taken from UCI repository [78]. Table 4.1 presents the number of samples and their attributes for each dataset used in the present work. The moderate size of datasets varies from 392 to 1030 samples. To balance the influence of each characteristic, all training and target set features are normalised. We divide the data into training and testing sets by $5-fold$ cross-validation during the regression model development, and the experiments are repeated from 100 to 1000 times. To validate the proposed ε-MDSVR, the widely used performance metrics are evaluated along with computation time using Linear and RBF kernel . The same results are compared and tested with the ϵ-SVR, Linear regression (LinReg) and Logistic Regression (LogReg) techniques also. The parameter, ϵ of the strict restoration adjustment is fixed at 0.1. The value C is fixed at 100, in all the experiments. The proposed algorithm is run on a 3.4 GHz Intel Core i7 2600 CPU with 8 GB RAM using MATLAB 9.2 platform.

Table 4.1: Benchmark datasets used in the present study

Dataset	Samples	Attributes
Auto MPG	392	7
Forest Fires	517	12
Energy Efficiency	768	8
Concrete Compressive Strength	1030	8

4.4.2 Performance evaluation

Mean square error (MSE) and the coefficient of determination (R^2) are the two popular metrics used for evaluating regression models. MSE in Equation (4.24) is defined as the average of squares of the errors that is used to check how close estimate values are to the actual values. The lower the MSE value, the better the model's forecasting performance. The R^2 in Equation (4.25) is calculated by dividing the sum of squares of residuals from the regression model by the total sum of squares of errors from the average model and then subtracting it from 1. The R^2, which ranges from 0 to 1, is a metric that can be used to assess the accuracy of predictions based on real data. The higher the R^2 value, the better the fit of the observations.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{X}_i - X_i\right)^2 \tag{4.24}$$

$$R^2 = \frac{n\sum_{i=1}^{n}\hat{X}_i X_i - \sum_{i=1}^{n}\hat{X}_i\sum_{i=1}^{n}X_i}{\sqrt{\left(n\sum_{i=1}^{n}\hat{X}_i^{\,2} - \left(\sum_{i=1}^{n}\hat{X}_i\right)^2\right)\left(n\sum_{i=1}^{n}X_i^{\,2} - \left(\sum_{i=1}^{n}X_i\right)^2\right)}} \tag{4.25}$$

$\hat{X}_i$ is the vector denoting values of n number of predictions and X_i is a vector representing n number of true values.

Figure 4.1 shows the line plot of MSE values versus number of iterations of proposed ε-MDSVR when compared with classical techniques such as ϵ-SVR over linear and RBF kernel functions, also with Linear and Logistic regression techniques. One can see that the proposed method has achieved reduced error rate when compared with given techniques on four datasets. This indicates that the ε-MDSVR works better than its counter parts. The same can be proved using Table 4.2, which summarizes the average MSE values of proposed and existing algorithms on four datasets. Dataset(n/m) indicates that dataset is having n samples and m attributes. It is observed that ε-MDSVR has given better results on the four datasets when compared with other methods.

The coefficient of determination, R^2 is used to validate the regression efficiency of algorithm. The low value of R^2 represents the weak relationship between the response y and the predictor x and high R^2 represents the strong relationship between the response y and the predictor x. Figure 4.2 depicts the line plot of R^2 values versus number of iterations of proposed ϑ-DWLSVR when compared with classical techniques such as ϵ-SVR over linear and RBF kernel functions, also with Linear and Logistic regression techniques. It can be seen that the proposed ε-MDSVR comparatively achieves higher values of R^2 than other techniques, which indicates better goodness of fit. Table 4.3 lists the average R^2 values of proposed and existing algorithms on four datasets. The results prove our point mentioned above. The average CPU time (seconds) of proposed and existing algorithms on each data set is also compared in Figure 4.3. It can be verified that ε-MDSVR costs less time than ϵ-SVR on most datasets and it is only slightly slower than other techniques on given datasets which can be verified in Table 4.4. We applied Wilcoxon Signed-Rank test to assess the statistical significance of the results obtained by proposed and existing algorithms against MSE, R^2 and CPU time. The obtained results are statistically significant with $\rho < 0.05$. Finally, the fitting quality of ε-MDSVR is much better and is more competitive when compared with existing techniques.

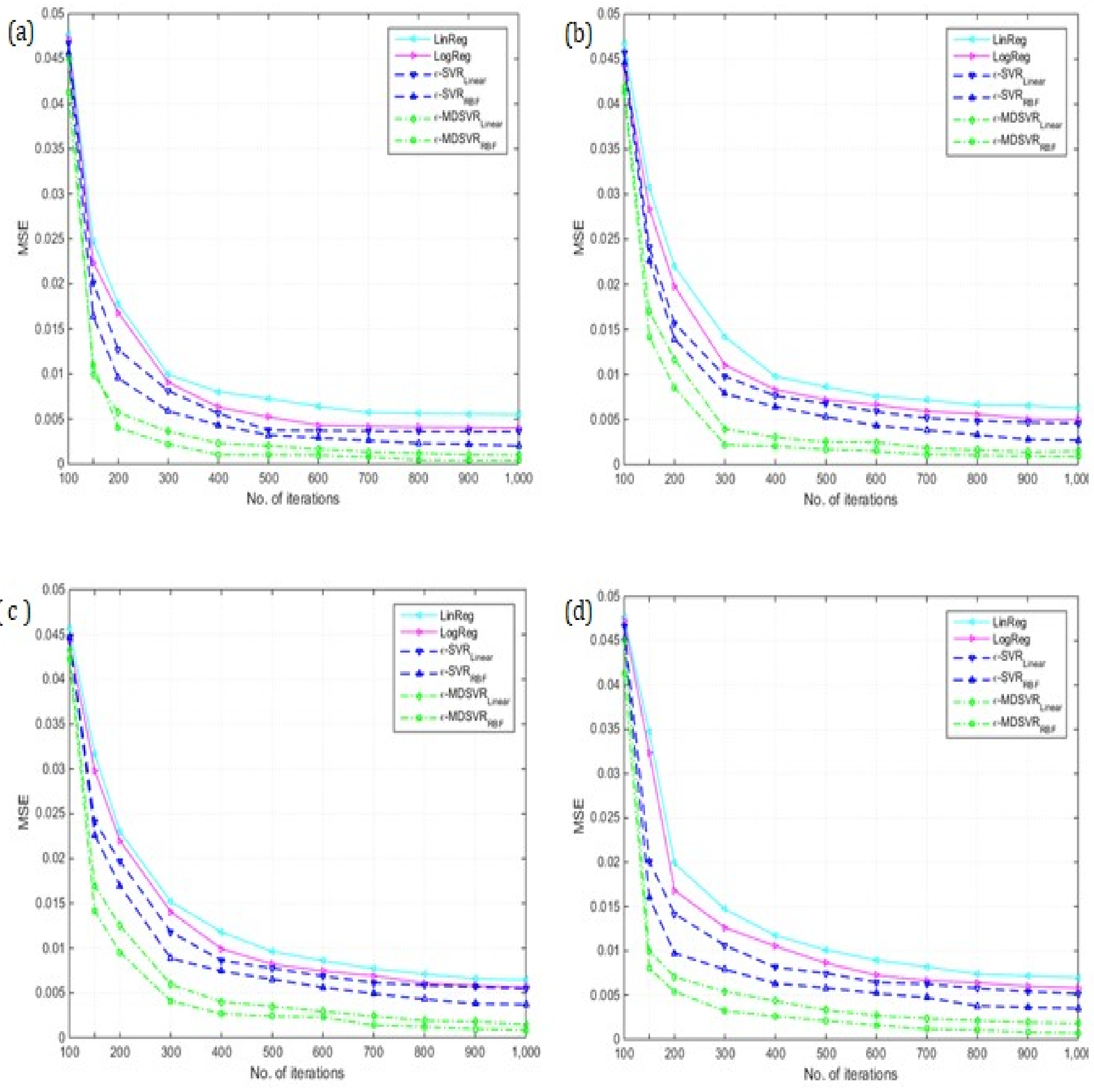


Figure 4.1: Line plots showing the MSE versus Number of iterations of four datasets evaluated using proposed ϵ-MDLSVR compared with ϵ-SVR using Linear and RBF kernels and also with Linear Regression and Logistic Regression. Figure 2(a): Auto MPG, Figure 2(b): Forest Fires, Figure 2(c): Energy effciency, Figure 2(d): Concrete compressive strength.

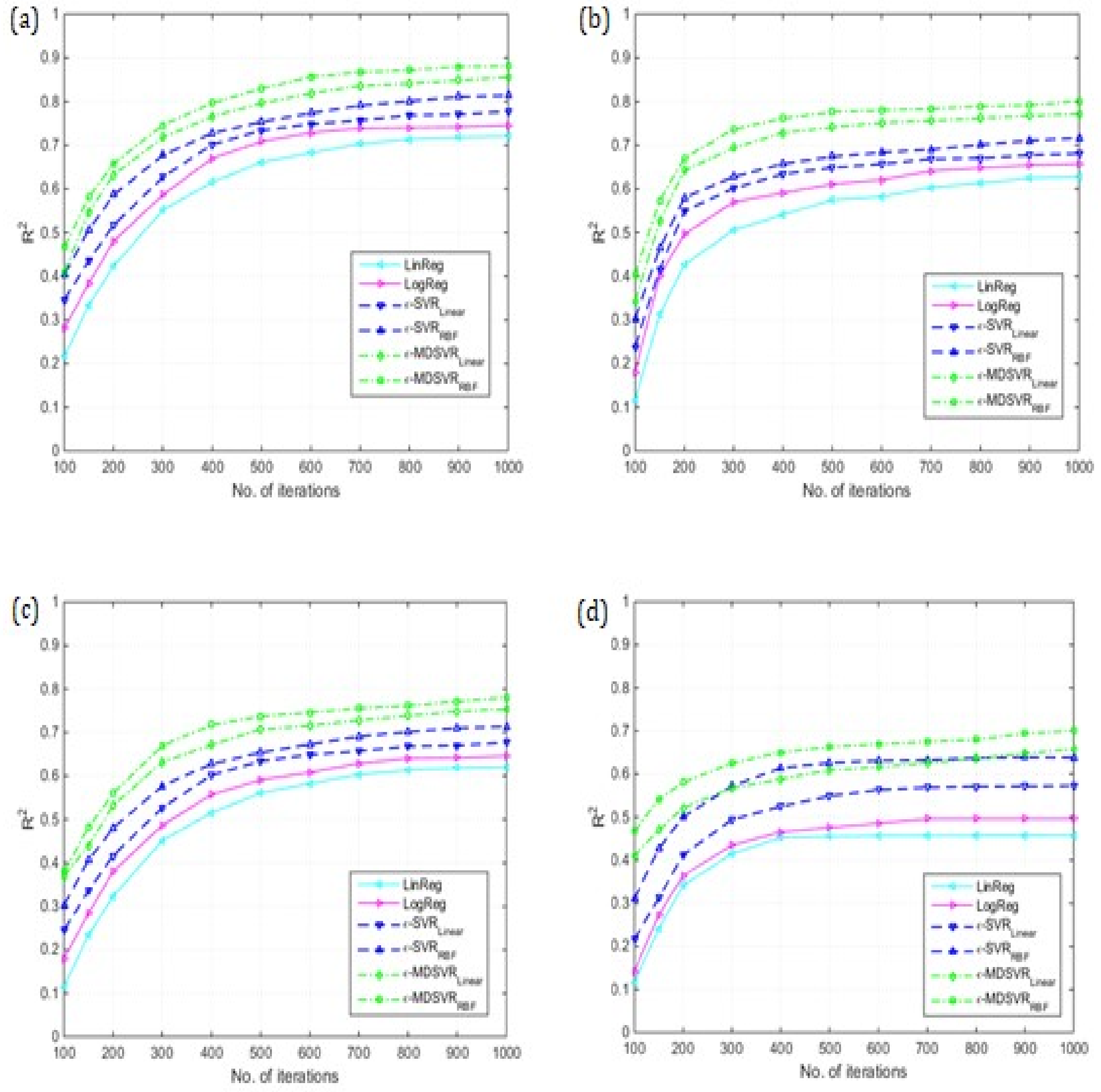


Figure 4.2: Line plots showing the R^2 versus Number of iterations of four datasets evaluated using proposed ϵ-MDLSVR compared with ϵ-SVR using Linear and RBF kernels and also with Linear Regression and Logistic Regression. Figure 4.2(a): Auto MPG, Figure 4.2(b): Forest Fires, Figure 4.2(c): Energy effciency, Figure 4.2(d): Concrete compressive strength.

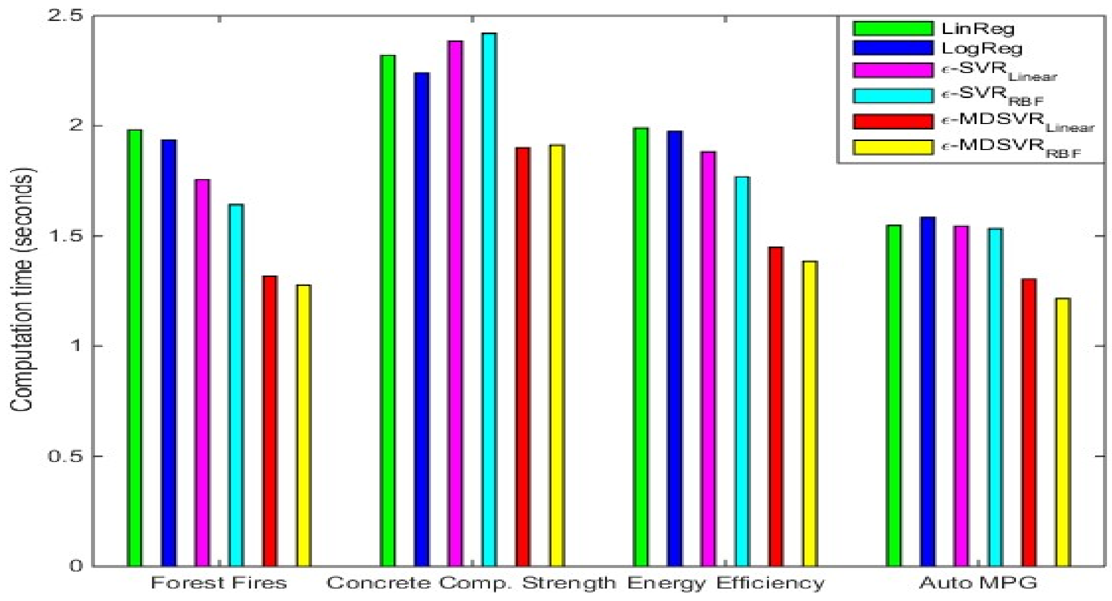


Figure 4.3: CPU time comparison between ε-MDSVR with ϵ-SVR and other Regression techniques with Linear and RBF kernels used in the model

Table 4.2: Validation of proposed ε-MDSVR using MSE values and compared with existing techniques such as ϵ-SVR, Linear and logistic regression.

Datasets/Algorithms	Linear Kernel		RBF kernel			
	ε-MDSVR	ε-SVR	ε-MDSVR	ε-SVR	LinReg	LogReg
Auto MPG (392/7)	0.0068	0.0103	0.0057	0.0081	0.0131	0.0116
Forest Fires (517/12)	0.0078	0.0123	0.0061	0.0101	0.0162	0.0145
Energy Efficiency (768/8)	0.0080	0.0122	0.0068	0.0107	0.0151	0.0133
Concrete compressive strength (1030/8)	0.0087	0.0133	0.0074	0.0117	0.0157	0.0145

4.5 Summary

This chapter proposed ε-MDSVR algorithm that utilizes the user-defined parameters c, k, and C, and improves the generalisation ability by finding a fair trade-off between the insensitive loss function and the quadratic loss function. To solve the optimization problem, we use a modified DCD approach. At each iteration, the proposed DCD technique constantly picks one variable for minimization while keeping the rest as constants. It chooses possibly the most effective variable to optimize at each iteration according to the possible decrease in values of the objective function of large margin distribution. To validate the proposed ε-MDSVR, UCI datasets are used and the performance metrics are evaluated along with computation time using Linear and RBF kernel. The

Table 4.3: Validation of proposed ε-MDSVR over linear and RBF kernel using R^2 values and compared with existing techniques such as ϵ-SVR, Linear and logistic regression.

Datasets/Algorithms	Linear Kernel		RBF kernel			
	ε-MDSVR	ε-SVR	ε-MDSVR	ε-SVR	LinReg	LogReg
Auto MPG (392/7)	0.733	0.652	0.766	0.694	0.576	0.618
Forest Fires (517/12)	0.680	0.585	0.715	0.618	0.502	0.551
Energy Efficiency (768/8)	0.639	0.552	0.669	0.594	0.476	0.513
Concrete compressive strength (1030/8)	0.577	0.486	0.631	0.566	0.391	0.420

Table 4.4: Validation of proposed ε-MDSVR over linear and RBF kernel using computation time in seconds and compared with existing techniques such as ϵ-SVR, Linear and logistic regression.

Datasets/Algorithms	Linear Kernel		RBF kernel			
	ε-MDSVR	ε-SVR	ε-MDSVR	ε-SVR	LinReg	LogReg
Auto MPG (392/7)	1.30	1.54	1.21	1.53	1.54	1.58
Forest Fires (517/12)	1.31	1.75	1.27	1.64	1.98	1.93
Energy Efficiency (768/8)	1.44	1.88	1.38	1.76	1.99	1.97
Concrete compressive strength (1030/8)	1.90	2.38	1.91	2.41	2.32	2.24

same results are compared and tested with the ϵ-SVR, Linear regression (LinReg) and Logistic Regression (LogReg) techniques also. The proposed algorithm with RBF kernel significantly achieves better prediction accuracy with MSE=0.0074, R^2=0.631 and performs well when compared to the classical SVR and other regression techniques. The results show that the proposed model gives good prediction accuracy and has a faster learning speed with low computational cost when compared to classic strategies. Thus, the algorithm is suitable for large-scale problems.

Chapter 5

Incremental ϑ-Support vector regression optimized with bounded estimation functions to handle noisy datasets

The chapter details the proposed ϑ-Support vector regression algorithm (optϑ-SVR) that is used to handle uncertain data along with distribution of data on decision curve by applying bounded functions to adjust the number of support vectors and errors. The proposed method formulates upper bound and lower bound functions on perturbed data. Two special adjustments to the support vectors and penalty parameters are used to enable the model to learn incrementally. The proposed method is implemented using Parkinson's disease dataset taken from PPMI repository. The results shows that the algorithm generates a more smoother regression curve and achieves better prediction accuracy compared to classic ϑ-SVR technique.
The main contributions of Proposed work are as follows:

- ϑ-Support vector regression algorithm optimized with distance weighted strategy (DWD) to define lower and upper bounds on perturbed data, is proposed.
- The proposed method formulates upper bound and lower bound functions on perturbed data.
- Two special adjustments to the support vectors and penalty parameters are used to enable the model to learn incrementally.
- The model is evaluated on Parkinson's disease dataset from PPMI repository.

- The performance measures of Mean square error (MSE), Regression Coefficient (R^2) and computation time are validated with the classical SVR.
- The results show that the proposed algorithm works better than the classical ϑ-SVR which affects the generalization performance and computational overheads.

5.1 Introduction

Incremental learning of kernelized SVR suffers with the curse of kernelization and cannot handle uncertain data [39]. When data is projected to high dimensional space, it is often that data is densely distributed on the boundary. Both ε-SVR and ϑ-SVR are vulnerable to the distribution of boundary data (support vectors). These support vectors are numerous in high dimensional space and all pile up at boundaries of the margin. when training sets with uneven sizes are used, the resulting function undesirably gets biased towards the majority class. Also, if the number of support vectors (SVs) increases in each iteration of the optimization, there will be a non-linear growth of model update time and prediction time with data size [40]. During learning, the optimization function critically depends on width of the margin and this influences the final estimation function count on the distribution of the support vectors. Recent advancement in SVM theory, however, revealed that aiming to maximize the minimum distance between instances to the boundary does not always lead to reduced generalisation error. But maximising the mean of the functional margin of whole data can improve the performance effectively by employing DWD strategy [41, 42]. By maximising the mean of the functional margin of all data, and making use of its distances to define the regression curve can improve the performance. Inspired by the idea of distance weighted strategy (DWD), we can adjust the bounds on support vectors of ϑ-SVR that allows the flexibility of specifying errors for uncertain data, thus improving the performance of ϑ-SVR on uncertain data.

The proposed method formulates upper bound and lower bound functions on perturbed data by making adjustments to the support vectors and penalty parameters which are used to enable the model to learn incrementally. If the value of w is determined, the ranking of all data points to the fitting surface with respect to the margin can be decided by functional margin. The proposed method uses distance weighted discrimination to reduce the distribution of data and defines the regression curve. To evaluate generalization performance, 10-fold cross validation is used and is implemented using Parkinson's disease data taken from PPMI repository. The model is evaluated on Mean square error (MSE), Regression Coefficient (R^2) and computation time using different kernels like linear, polynomial, sigmoid, radial basis function (RBF), Logistic kernel and the results are validated with the classical SVR. The algorithm generates a more smoother regression curve and achieves better prediction accuracy with MSE=0.131 and R^2=0.758 in less computation time of 2.26 sec, compared to classical ϑ-SVR.

The rest of the content of this chapter is organized as follows. Section 5.2 presents the basic equations used to derive the proposed model. Implementation details of the proposed algorithm are described in Section 5.3. Experiment results and validation of the proposed model and its behaviour against other recent existing research is discussed in Section 5.4. The summary of the Chapter is given in section 5.5.

5.2 Preliminaries

The task of ML is to design a system that gives good generalization performance. This involves searching for a required classification model over a given dataset by optimizing its objective function on the model's parameter space [59]. This classical search can be computationally intensive when training data is continuously coming, as the model needs to tune hyper-parameters of the objective function at each iteration [58]. Thus, one requires a learning paradigm that can process sequential data in a streaming fashion. Due to the inadequate efficacy of standard easy to implement linear algorithms, advanced non-linear time-series prediction techniques such as neural networks (NN) and support vector machines (SVM) are introduced gradually in machine learning communities. Amongst these, SVM gained profound attention for several reasons [79, 80]. The next section gives the basic formulation of SVR and consequently shows how it is used to handle continuous data.

5.2.1 ϑ-SVR formulations

SVM solves regression problems using ε-insensitive loss function (ε : error deviation) generally referred to as ε-SVR [18]. ε-SVR aims to find a function whose deviation is not more than ε, thus forming the ε-tube, to fit all the training data. Let $T = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$ be a training set of n samples, where $x_i \in \mathbb{R}^m$ are the input values and corresponding target values,

$y_i \in \mathbb{R}$. The objective function is:

$$f\left(x\right) = w \cdot \phi\left(x\right) + b \tag{5.1}$$

The premise that there exists a function that approximates each data pair (x_i, y_i) with a suitable ε accuracy makes this convex optimization problem viable. To regulate the number of support vectors and errors, a parameter ϑ, $(0 \leq \vartheta \leq 1)$ is added to the original ε-SVR. The fraction of margin errors and support vectors are bound by maximum and minimum value of ϑ. The objective function $f\left(x\right)$ of ϑ-SVR is represented by the following constrained minimization problem in Equation (5.2).

$$\min_{w,\varepsilon,\xi_i^*} \frac{1}{2} \parallel w \parallel^2 + C \cdot \left(\vartheta\varepsilon + \frac{1}{n} \sum_{i=1}^{n} \left(\xi_i + \xi_i^*\right) \right) \tag{5.2}$$

$$\begin{aligned} \text{s.t} \quad & \left(w \cdot \phi\left(x_i\right) + b\right) - y_i \leq \varepsilon + \xi_i, \\ & y_i - \left(w \cdot \phi\left(x_i\right) + b\right) \leq \varepsilon + \xi_i^*, \\ & \xi_i, \xi_i^* \geq 0,\ \varepsilon \geq 0,\ i = 1, 2, \cdots, n. \end{aligned}$$

The penalty factor C reflects the trade-off between error and margin i.e., the optimization criterion penalizes data points whose y-value differs from $f(x)$ by more than ε. Lagrangian multipliers with positive and non-zero of α_i and α_i^* are called the support vectors (SVs). After applying Lagrangian multipliers, the final objective function $f\left(x\right)$ is given in the following Equation (5.3).

$$f\left(x\right) = \sum_{i=1}^{n} \left(\alpha_i - \alpha_i^*\right) K\left(x_i, x\right) + b \tag{5.3}$$

The decision functions obtained by two methods are identical if the values of parameter C are same, and the parameter ε has a relationship with the parameter ϑ.

5.2.2 DWD learning

SVM is evaluated using a minmax optimization formulation, focusing mainly on number of SVs i.e., samples that resides right on the separating hyperplane. However, these SVs tend to pile up at the boundaries of margin when data is projected into higher dimensions, which diminishes the generalization performance of SVM. Data piling and over-fitting issues in SVM can be solved by using DWD technique [42, 47]. DWD permits a larger number of data points to have a direct impact on the weight vector w, and it emphasises training instances that are close to the hyperplane. DWD replaces the SVM criterion of maximising the lowest margin with one that maximises the mean of the functional margin. By maximising the minimal margin between the training points and the classification boundary, the SVM classifier seeks to minimise generalisation error. In an SVM classification problem, minimizing of margin is equivalent to minimising an upper constraint on the Vapnik–Chervonenkis (VC) dimension of the classifying hyperplane. DWD aims to maximise the distance of every observation to the separating hyperplane by minimizing the sum of the inverse of every residual. The functional margin in SVR can be described as a difference/adjusted distance between the real values and estimated values of objective function.

DWD denotes the functional margin u_i for the i^{th} data point from separating hyperplane and is given as:

$$u_i = y_i(w \cdot \phi(x_i) + b) \tag{5.4}$$

Let r_i be the adjusted distance of the i^{th} data point to the separating hyperplane by allowing some error vector ζ_i which is given as

$$r_i = y_i(w \cdot \phi(x_i) + b) + \xi_i \tag{5.5}$$

Therefore, the solution of DWD will be

$$\underset{w,b,\xi_i}{min} \sum_{i=1}^{n} (\frac{1}{r_i} + C\xi_i) \tag{5.6}$$

s.t $\qquad r_i \geq 0, \qquad \xi_i \geq 0, \qquad \| w \|^2 \leq 1, \qquad i = 1, \cdots, n.$

When $\xi_i = 0$ and $y_i(w \cdot \phi(x_i) + b) > 0$, $r_i = y_i(w \cdot \phi(x_i) + b)$, is the positive distance from each data vector to the separating hyperplane due to Equation (5.4) and Equation (5.5). Thus, $\sum_{i}^{n} \frac{1}{r_i}$ defines a different notion of gap between classes from that of SVM ($\frac{2}{\|w\|}$). If a positive distance $y_i(w \cdot \phi(x_i) + b)$ is not achievable for a data vector, then a positive slack variable ξ_i is added to make r_i positive. However, the value of ξ_i corresponds to the amount of misclassification for the i^{th} vector, and hence in order to minimize the misclassification, we must control $\sum_{i}^{n} \xi_i$ in the objective function. Using this formulation of DWD and combining it with the SVM method, the underlying DWD loss can be obtained as follows: For each i, the term $(\frac{1}{r_i} + C\xi_i)$ in the objective function of Equation (5.6) can be minimized over $\xi_i x$. the optimization problem of SVM over w and b is given as:

$$\min_{w,b} \quad \sum_{i}^{n} V_c(y_i(w \cdot \phi(x_i) + b)) \quad s.t \quad \|w\|^2 \leq 1 \tag{5.7}$$

where, DWD loss function is defined as:

$$V_C(u) = \begin{cases} 2\sqrt{C} - Cu & \text{if } u \leq \frac{1}{\sqrt{C}} \\ 1/u & \text{otherwise.} \end{cases} \tag{5.8}$$

The two key observations in naive SVM classification is: 1) The sum of inverse distance, ($\sum_i r_i^{-1}$) is to be maximised to distinguish between two classes. 2) A measure of misclassification, $\sum_i \xi_i$ that is to be minimized. Distance-weighted Support Vector Machine method possesses the merits of both SVM and the DWD and can overcome the datapiling and over-fitting issues of SVM and DWD and it also follows faster training approach for large scale datasets [42, 47, 48].

5.3 Proposed ϑ-PSVR

ϑ-SVR uses the same SVM theory to the regression problems. As in SVM, the fitting curve is also affected by the distribution of boundary data i.e., data piling and over-fitting issues. The proposed method uses ϑ-SVR trained over incremental learning and optimized to handle noisy data to enhance the performance. The proposed incremental ϑ-PSVR is designed to deal with the noisy sample regression problems. To evaluate generalization performance, 10-fold cross validation is used.

5.3.1 ϑ-SVR on perturbed data

The proposed algorithm addresses the key issues of data piling and over-fitting effect, due to uncertain data present in classical ϑ-SVR. This effects the generalization performance and computational overheads. The proposed method formulates upper bound and lower bound functions on perturbed data. Two special adjustments to the support vectors and penalty parameters are used to enable the model to learn incrementally. If the value of w is determined, the ranking of all data points to the fitting surface with respect to the margin can be decided by functional margin. To handle the input data with noise, two bounded estimation functions $f_1(x) = w_1.x + b_1$ and $f_2(x) = w_2.x + b_2$ are defined. Thus, the Regression curve is given as $f(x) = \frac{1}{2}[f_1(x) + f_2(x)]$.

Let the data point be (x_i, y_i), where x_i and y_i are perturbed by noise of δ_i, $\check{\delta}_i$. Then the perturbed data is represented as:

$x_i + \delta_i$, $(\|\delta_i\| \leq \tau | \tau > 0)$; $\quad y_i + \check{\delta}_i = [u, v]$, $(\|\check{\delta}_i\| \leq \check{\tau} | \check{\tau} > 0)$.

We get, $w.\phi(x_i + \delta) = (w.\phi(x_i) + w.\phi(\delta))$; $|w.\phi(\delta)| \leq \|w\|.\|\delta\| \leq \tau\|w\|$. Hence, According to eq. (5.2), modified ϑ-SVR bounded functions are:

$$\min_{w_1, b_1, \xi_{1i}} \frac{n}{2} \|w_1\|^2 + C_1 \left(\vartheta_1 b_1 n + \sum_{i=1}^{n} \xi_{1i} \right) \tag{5.9}$$

s.t. $(w_1 \cdot \phi(x_i) + b_1) + \tau\|w_1\| \geq u - \xi_{1i}, \quad \xi_{1i} \geq 0.$

$$\min_{w_2, b_2, \xi_{2i}} \frac{n}{2} \|w_2\|^2 + C_2 \left(\vartheta_2 b_2 n + \sum_{i=1}^{n} \xi_{2i} \right) \tag{5.10}$$

s.t. $(w_1 \cdot \phi(x_i) + b_2) + \tau\|w_2\| \geq v + \xi_{2i}, \quad \xi_{2i} \geq 0. \quad i = 1, 2, \cdots, n.$

C_1, C_2 are penalty factors and ξ_{1i}, ξ_{2i} are the slack variables to measure the amount of difference between the estimated and the target value. $\vartheta \in (0, 1)$ controls minimization of b_1, b_2 and the errors. let $Q_{ij} = \frac{1}{n} K(x_i, x_j) = \frac{1}{n}(\phi(x_i) + \tau) \cdot (\phi(x_j) + \tau)$; then, the dual problem of eq. (5.9) can be written:

$$\min_{\alpha} \frac{1}{2} \sum_{i,j=1}^{n} \alpha_{1i}\alpha_{1j}, Q_{ij} - \sum_{i=1}^{n} y_{1i}\alpha_{1i} \tag{5.11}$$

s.t. $\sum_{i=1}^{n} \alpha_{1i} = C_1 \vartheta_1 n, \quad 0 \leq \alpha_{1i} \leq C_1, \quad i = 1, \ldots, n.$

5.3.2 Incremental learning

If a new sample (x_k, y_k) arrives in the training data set T, there exists an increment in the extended training sample set say S, which can be defined as:

$S = S^- \cup S^+$, $S^- = \{x_{1i}, y_{1i}, z_{1i} = -1\}_{i=1}^{n}$, $\quad S^+ = \{x_{1i}, y_{1i}, z_{1i} = +1\}_{i=1}^{n}$,

z_i be the label of training sample (x_{1i}, y_{1i}) Initially, the weights of each data point in S are set to zero before adding new sample. If this initialization violates the KKT conditions after adding new sample, weight adjustments will become mandatory. The incremental ϑ-SVR algorithm continuously updates the weights when KKT conditions are not held for any new sample added to the T. Thus, the dual form of the ϑ-SVR in Equation (5.11) can be further represented as:

$$\min_{\alpha} \frac{1}{2} \sum_{i,j=1}^{2n} \alpha_{1i}\alpha_{1j}, Q_{ij} \tag{5.12}$$

s.t. $\sum_{i=1}^{2n} z_{1i}\alpha_{1i} = 0, \quad \sum_{i=1}^{2n} \alpha_{1i} = 2C_1\vartheta_1 n, \quad 0 \leq \alpha_{1i} \leq C_1, i = 1, \ldots, 2n$

Apply Lagrangian multipliers μ, ρ;

$$\min_{0\leq\alpha_{1i}\leq C_1} w = \frac{1}{2}\sum_{i,j=1}^{2n} \alpha_{1i}\alpha_{1j}Q_{ij} + \mu\left(\sum_{i=1}^{2n} z_{1i}\alpha_{1i}\right) + \rho\left(\sum_{i=1}^{2n}\alpha_{1i} - 2C_1\vartheta_1 n\right) \tag{5.13}$$

first order derivative of w leads to following KKT conditions
$\frac{\partial w}{\partial \mu} = \sum_{i=1}^{2n} z_{1i}\alpha_{1i} = 0; \quad \frac{\partial w}{\partial \rho} = \sum_{i=1}^{2n} \alpha_{1i} = 2C_1\vartheta_1 n$

$$\forall i \in S : h(x_{1i}) = \frac{\partial}{\partial \alpha_{1i}} = \sum_{j=1}^{2n}\left(Q_{ij}\alpha_{1i} + z_{1i}\mu + \rho\right); \begin{cases} \geq 0 & \text{if } \alpha_{1i} = 0 \\ = 0 & \text{if } 0 < \alpha_{1i} < C_1 \\ \leq 0 & \text{if } \alpha_{1i} = C_1. \end{cases} \tag{5.14}$$

Based on $h(x_{1i})$, T is partitioned into three independent sets.
$S_S = \{i : h(x_{1i}) = 0, \quad 0 < \alpha_{i1} < C_1\}$ vectors on the curve
$S_E = \{i : h(x_{1i}) \leq 0, \quad \alpha_{i1} = C_1\}$ vectors deviated from the curve
$S_R = \{i : h(x_{1i}) \geq 0, \quad \alpha_{i1} = 0\}$ vectors covered by bounded error

The weights of new sample (x_c, y_c) are set to 0; $\alpha_c = 0$, and then α_c value gradually changes at each iteration by migrating samples from one set to another until all KKT conditions are satisfied. The incremental relation between $\Delta h(x_{1i})$, $\Delta\alpha_{1i}$, $\Delta\mu$ and $\Delta\rho$ is :

$$\Delta h(x_{1i}) = \sum_{j\in D_S} Q_{ij}\Delta\alpha_{1j} - Q_{ic}\Delta\alpha_c + z_{1i}\Delta\mu + \Delta\rho = 0 \tag{5.15}$$

$\sum_{j\in D_S} z_{1j}\Delta\alpha_{1j} + z_{1c}\Delta\alpha_{1c} = 0$, α_c is the coefficient being incremented.
The linear relationship between $\Delta h(x_{1i})$ and $\Delta\alpha_c$ is:

$$\Delta h(x_{1i}) = \left(\sum_{j\in D_S} \beta_j^c Q_{ij} + z_{1i}\beta_\mu^c + \beta_\rho^c + Q_{ic}\right)\Delta\alpha_{1c} \equiv \gamma_{1i}^c\Delta\alpha_{1c}, \forall i \in S_S \tag{5.16}$$

where, β_j^c stands for dimensions corresponding to S_s matrix, β_μ^c are the vectors corresponding to S_E matrix, β_ρ^c are the vectors corresponding to S_R matrix.

5.3.3 Optimization of ϑ-PSVR

Optimal solution of the dual minimization problem in Equation (5.12) is resolved when $Q = \frac{n}{n+1}Q$.

$$\min_{\alpha} \quad \frac{1}{2(n+1)} \sum_{i,j=1}^{2n} \alpha_{1i}\alpha_{1j}Q_{ij} \tag{5.17}$$

s.t $\quad \sum_{i=1}^{2n} \alpha_1 i = 0, \quad \sum_{i=1}^{2n} \alpha_{1i} = C_1\vartheta_1 n, \quad 0 \le \alpha_{1i} \le C_1, \quad i = 1, \cdots, 2n$

The primal of the Equation (5.17) can be represented as

$$\min_{w,b,\varepsilon,\xi_i,\xi_i^*} \quad \frac{n+1}{2} \parallel w \parallel^2 + C.\left(\vartheta\varepsilon n + \sum_{i=1}^{n}(\xi_i + \xi_i^*)\right) \tag{5.18}$$

s.t $\quad (w \cdot \phi(x_i) + b) - y_i \le \varepsilon + \xi_i,$

$\quad y_i - (w \cdot \phi(x_i) + b) \le \varepsilon + \xi_i^*,$

$\quad \xi_i, \xi_i^* \ge 0, \ i = 1, 2, \cdots, n.$

According to KKT conditions,

$$w = \frac{1}{n+1} \sum_{i=1}^{2n} \alpha_{1i}\phi(x_i) \tag{5.19}$$

The functional margin of ϑ-SVR

$$\gamma = (y_i - (w.\phi(x_i) + b))^2, \quad i = 1, \cdots, n. \tag{5.20}$$

The optimal solution w in Equation (5.19) can be rewritten by considering mean of the functional margin is given in Equation (5.21).

$$w = \frac{1}{n+1} \sum_{i=1}^{2n} (\alpha_i - \alpha_i^*)X_i + u \tag{5.21}$$

$X^T w = X^T(X(\alpha_i - \alpha_i^*) + u)$

$X^T w = X^T X(\alpha_i - \alpha_i^*) \quad i.e., X.u = 0$

$X^T w = G(\alpha_i - \alpha_i^*)$

$w^T w = (\alpha_i - \alpha_i^*)^T X^T(\alpha_i - \alpha_i^*)X$

$w^T w = (\alpha_i - \alpha_i^*)^T G(\alpha_i - \alpha_i^*)$

Where, G is the kernel matrix, written in Equation (5.22)

$$G = X^T X \tag{5.22}$$

The adjusted distance between the sample point and the fitting curve is the functional margin. After determining the value of w, the functional margin can be used to rank all sample points in relation to the fitting surface in terms of distance.

Applying optimization technique to the minimization problem in Equation (5.18) leads to the following solution such as,

$$\min_{w,\xi,\xi^*} \frac{1}{2} \parallel w \parallel^2 + \frac{\lambda_1}{l}(w^T \cdot X^T w - 2(XY)^T w) + C\vartheta\varepsilon n \sum_{i=1}^{n}(\xi_i + \xi_i^*) \tag{5.23}$$

The above equation can be further represented as

$$\min_{\alpha,\xi,\xi^*} \quad \frac{1}{2}(\alpha - \alpha^*)^T Q(\alpha - \alpha^*) + P^T(\alpha - \alpha^*) + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \tag{5.24}$$

s. t $\qquad y_i - (\alpha - \alpha^*)^T G_i \leq \varepsilon + \xi_i, \qquad \xi_i, \xi_i^* \geq 0, \qquad i = 1, \cdots, n.$
Where, $Q = Q\lambda_1, G^T \frac{G}{l} + G \quad P = -2\lambda_1 G \frac{Y}{l}$
The Equation (5.24) can be transformed to dual form using the Lagrange multipliers β', η'. To hold the KKT conditions, partial derivatives w.r.to α_i and ξ are set to zero.

$$\min_{\beta'} \quad f(\beta') = \frac{1}{2}(\beta')^T + \left(\frac{\lambda_1}{n}H_e - e\right)^T \beta' \tag{5.25}$$

s.t $\qquad 0 \leq \beta' \leq C, \qquad i = 1, \cdots, n \qquad \beta' = [\beta, \beta^*]$
where, $H = GQ^{-1}G$ Q^{-1} refers to the inverse matrix of Q and e stands for all-one vector. Minimize β_i by keeping the other $\beta_{i\neq j}$ as constants, one needs to solve the following sub-problem,

$$\min_{t} \quad f(\beta' + t_{e_i})\text{s.t} \quad 0 \leq (\beta' + t) \leq C \tag{5.26}$$

where $\mathbf{e}_i$ denotes the vector with 1 in the i^{th} coordinate and 0's elsewhere. By considering the $f(\beta' + t_{e_i})$ as a simple quadratic function of t and $0 \leq (\alpha_i - \alpha_i^*) \leq C$, the Equation (5.25) leads to a closed-form solution,

$$\begin{cases} [\bigtriangledown f(\beta')]_k \leftarrow \vartheta\epsilon k + (G(\alpha-\alpha^*) - y_k), \\ \qquad if \quad k = 1, \cdots, n. \\ [\bigtriangledown f(\beta')]_k \leftarrow \vartheta\epsilon k - (G(\alpha-\alpha^*) - y_{k-n}), \\ \qquad if \quad k = n+1, \cdots, 2n. \end{cases} \tag{5.27}$$

$$\beta'_k \leftarrow min(max\left(\beta'_k - \frac{[\triangledown f(\beta')]_k}{h_{kk}}, 0\right), C\vartheta k); \tag{5.28}$$

According to Equation (5.28), the prediction coefficients $(\alpha - \alpha^*)$ from the optimal β'_k are obtained as:

$$(\alpha - \alpha^*) = Q^{-1}G\left(\frac{\lambda_1}{n} + (\beta - \beta^*)\right) \tag{5.29}$$

Therefore, final fitting function can be calculated by:

$$f(x) = \frac{\mid W \cdot \phi(x_i) - y_i \mid}{\parallel w \parallel}; \tag{5.30}$$

After updating the variables, re-compute the inverse matrix $\check{R}$ for the next round of initial adjustments.

$$\check{R} \leftarrow \begin{bmatrix} \check{R} & 0 \\ 0 & 0 \end{bmatrix} + \frac{1}{\bar{\gamma}} \begin{bmatrix} \beta_{b'} \\ \beta_{\epsilon'} \\ \beta_{S_S} \\ 1 \end{bmatrix} \cdot \begin{bmatrix} \beta_{b'} \\ \beta_{\epsilon'} \\ \beta_{S_S} \\ 1 \end{bmatrix}^T \tag{5.31}$$

To obtain a better trade-off between the distribution of the complete data and the distribution of support vectors, we maximise the lowest margin while simultaneously maximising the margin distribution. The proposed ϑ-PSVR takes into account the influence of all data on the fitting surface, as this is more representative of the internal data distribution. A step by step procedure of above proposed method is given in Algorithm 5.1.

Algorithm 5.1 : Kernel ϑ-PSVR

```
Input: Dataset $X, y$ , $\lambda_1$ , $C$ , $\varepsilon$, $\vartheta$
Output: $S_S, S_E, S_R, f(x)$
Initialization: $\beta' = 0$ , u = 0 ; $(\alpha - \alpha^*)$ = $\frac{2\lambda_1}{n} Q^{-1}Gy$ ,$A = Q^{-1}G$, $h_{kk} = e_k^T GQ^{-1}Ge_k$ ,$g = \frac{n}{n+1}g$,$b' = \frac{n}{n+1}b'$, $\epsilon'$ = $\frac{n}{n+1}\epsilon'$ ,$\eta$ = $\frac{n}{n+1}\eta$
 1: for  iter $= 1, \cdots, MaxIter$  do
 2: for  $k = 1, \cdots, 2n$  do
 3: $\begin{cases} [\bigtriangledown f(\beta')]_k \leftarrow \vartheta\epsilon k + (G(\alpha - \alpha^*) - y_k); & if \quad k = 1, \cdots n. \\ [\bigtriangledown f(\beta')]_k \leftarrow \vartheta\epsilon k - (G(\alpha - \alpha^*) - y_k); & if \quad k = n+1, \cdots, 2n. \end{cases}$
 4: $\beta_k'^{old} \leftarrow \beta_k'$
 5: $\beta_k' \leftarrow min(max\left(\beta_k' - \frac{[\bigtriangledown f(\beta')]_k}{h_{kk}}, 0\right), C\vartheta k)$
 6: for $i = 1, \cdots, n$ do
 7: while $\eta \neq 1$ do
 8: $\begin{cases} (\alpha - \alpha^*) \leftarrow (\alpha - \alpha^*) + (\beta_k' - \beta_k'^{old})Ae_K; & if \quad k = 1, \cdots, n. \\ (\alpha - \alpha^*) \leftarrow (\alpha - \alpha^*) - (\beta_k' - \beta_k'^{old})Ae_K; & if \quad k = n+1, \cdots, 2n. \end{cases}$
 9: Weight updation
10: $w_i$ = $X_i(\alpha_i - \alpha_i^*) + u$
11: margin function $\gamma$
12: $\gamma = (w_i \cdot \phi(X_i) + b') - y_i)^2$
13: fitting curve f(x)
14: $f(x) = |(w \cdot \phi(x_i) - y_i)| / \| w \|$
15: update $\eta, g, \alpha, b', \varepsilon', S_S, S_E, S_R, \check{R}$
16: end while
17: end for
18: end for
19: if $\beta'$ converges then break
20: end if
21: end for
```

5.4 Experimental Results

The proposed ϑ-PSVR method is implemented using Parkinson's disease dataset (PD) taken from PPMI repository (http://www.ppmi-info.org/data). Further details of data used in the experiment are described in Section 3.4. The number of samples used in this analysis are n=600 with eleven predictors. The experiments are conducted for classification as well as for Regression. The classification is done by using class labels as (y_{+1}=PD, y_{-1}=Healthy), Regression estimation is done by considering UPDRS values of PD dataset as in Section 3.4. The optimal hyper parameters values of the model are obtained through 10-fold grid search cross validation and the experiments are repeated from 100 to 1000 times. For implementation of proposed algorithm, we set optimal hyper parameter values as $\varepsilon = 2e^{-5}$, $C = 10$ and T_e(tolerance-error) = $1e^{-6}$. To evaluate generalization performance of the model, LOOCV is implemented. LOOCV utilizes a single sample of dataset for testing and rest are used for training. This process is repeated for every sample in the data. The model is evaluated on Mean square error (MSE), Regression Coefficient (R^2) and computation time using different kernels like linear, polynomial, sigmoid, radial basis function (RBF), Logistic functions and the results are validated with the classical ϑ-SVR. The proposed algorithm is run on a 3.4 GHz Intel Core i7 2600 CPU with 8 GB RAM using MATLAB 9.2 platform.

The proposed algorithm minimizes the classification error by maximising the ε-insensitivity zone and produces sparser representation of support vectors. The contour plots from Figure 5.1 to Figure 5.3 shows the the class posterior probabilities of classical ϑ-SVR and proposed ϑ-PSVR. The contour plots display the separating hyperplane (black solid line) with ε-insensitivity zone (dashed lines). It is observed that some of the data points are placed in common space of both the margin planes, which represent the error rate of regression depicted as 'x' marker shape. In summary, the hyperplane defined by w and β is an auxiliary one, useful for finding the best direction towards the objective function with the intercept b which leads to good performance. When compared to the ϑ-SVR model, the ϑ-PSVR model has achieved considerable margin distance (dotted lines) between two classes of data for different kernels, and the number of error vectors (denoted by x) is drastically reduced. At the same time, when compared to other kernels, the proposed model with RBF kernel has shown a high margin distribution with low error as given in Figure 5.1, Figure 5.2 and Figure 5.3. As a result of the foregoing discussion, it can be concluded that the proposed ϑ-PSVR model outperforms and improves the generalisation power of the standard ϑ-SVR in classifying the dataset.

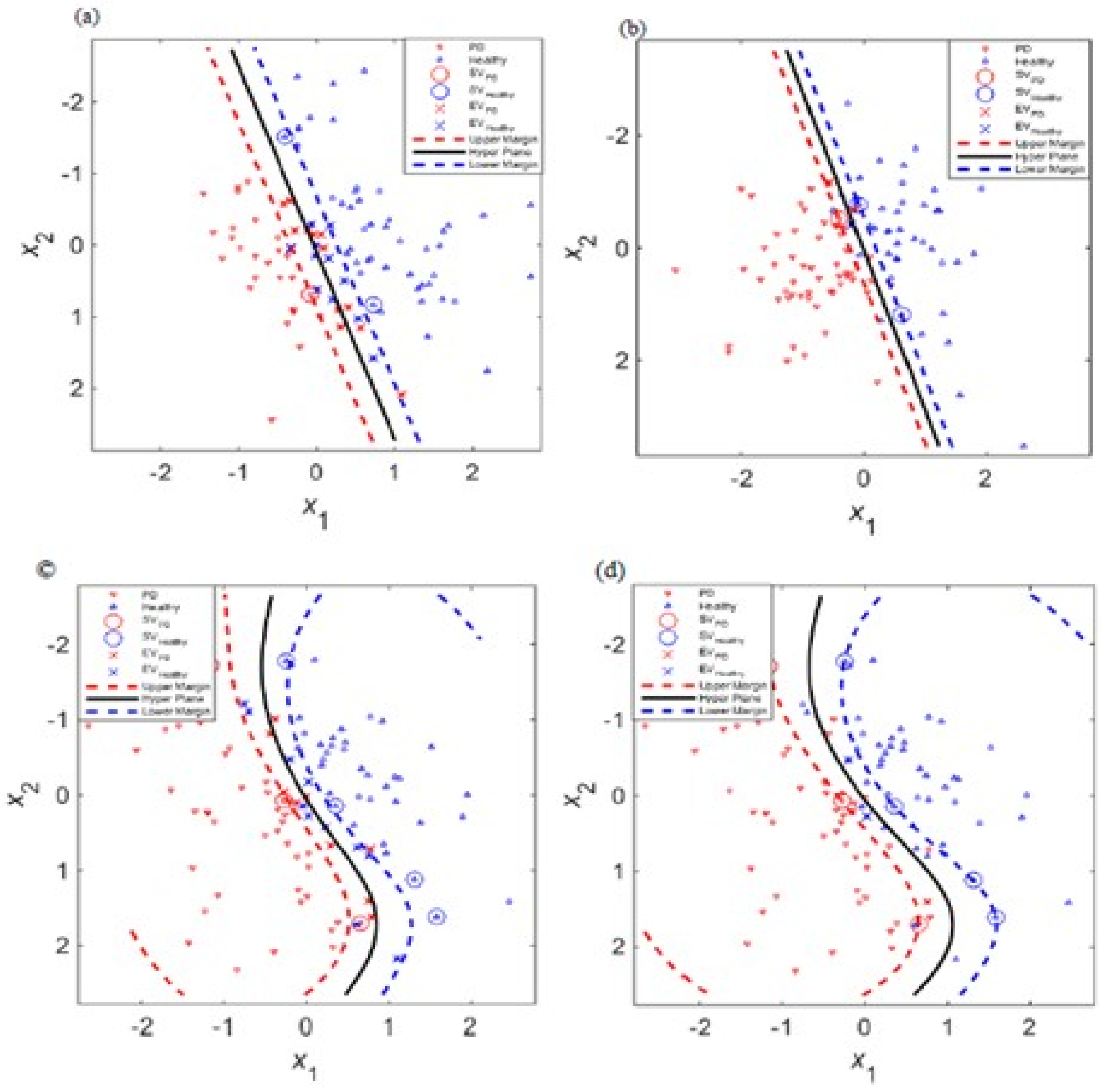


Figure 5.1: Contour plots showing the class separation of data using classical ϑ-SVR model with Linear Kernel in Figure 5.1(a) and RBF kernel in Figure 5.1(c), and proposed ϑ-PSVR with Linear Kernel in Figure 5.1(b) and RBF Kernel in Figure 5.1(d). SV=support vectors and EV=error vectors.

To show the effectiveness of the proposed algorithm in terms of computation time, we perform experiments on same dataset using proposed model and classical ϑ-SVR for each of the kernel functions and provided the comparison of the two algorithms based on CPU time (seconds). The proposed model with incremental learning with gradient descent estimation of weights significantly reduces the computation time when compared with non incremental classical ϑ-SVR.

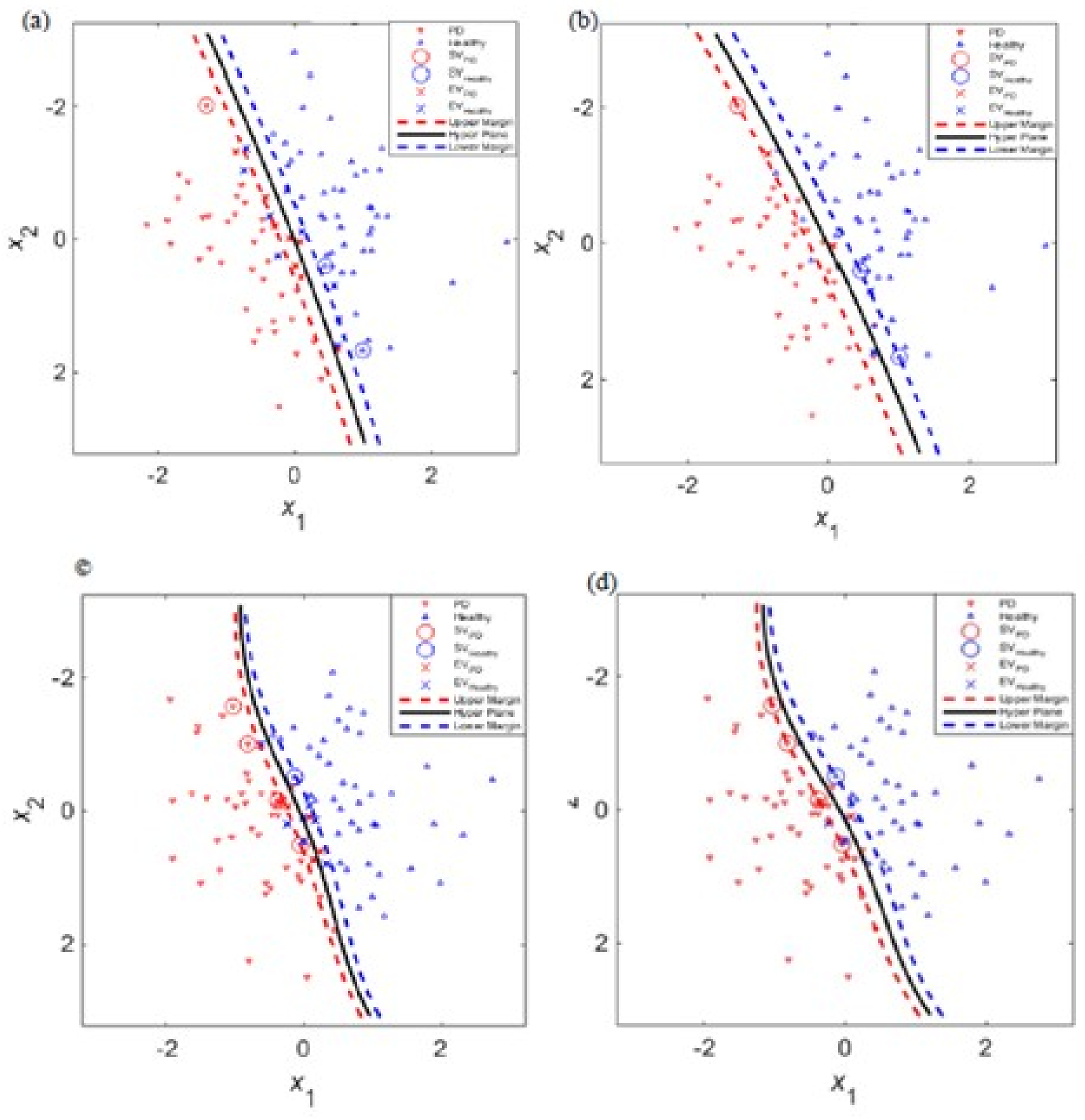


Figure 5.2: Contour plots showing the class separation of data using classical ϑ-SVR model with sigmoid Kernel in Figure 5.2(a) and polynomial^4 in Figure 5.2(c), and proposed ϑ-PSVR with sigmoid Kernel in Figure 5.2(b) and polynomial^4 in Figure 5.2(d). SV=support vectors and EV=error vectors.

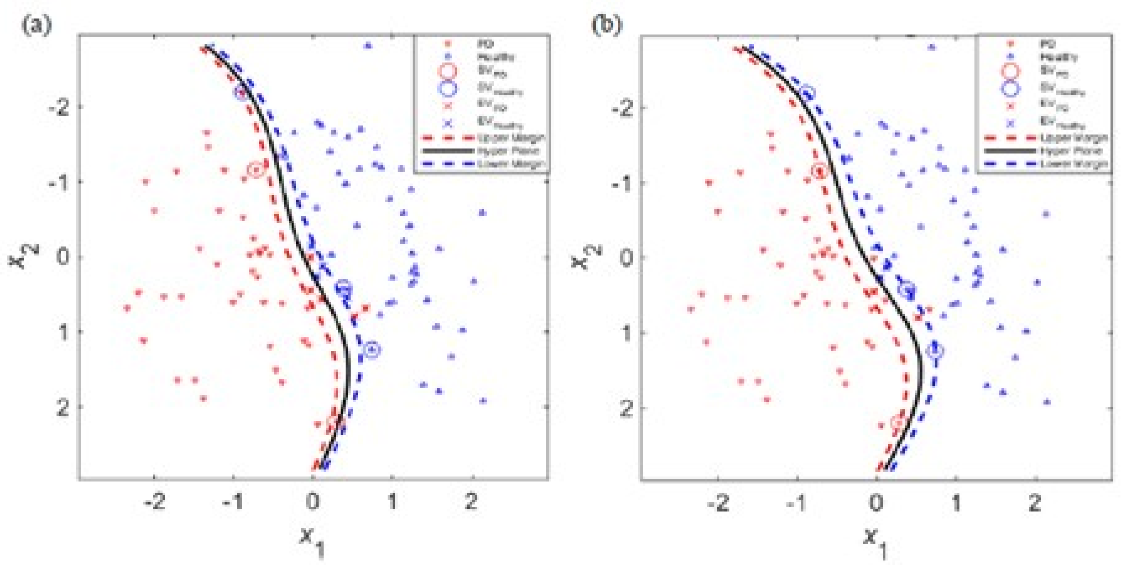


Figure 5.3: Contour plots showing the class separation of data using classical ϑ-SVR model in Figure 5.3(a) and proposed ϑ-PSVR in Figure 5.3(b) with logistic Kernel. SV=support vectors and EV=error vectors.

MSE values are used to check how close estimated values are to actual values and a smaller value of MSE leads to a better forecasting performance of the system. Figure 5.4 shows the line plot of MSE values versus number of iterations of proposed ϑ-PSVR over linear, RBF, polynomial, sigmoid and Logistic functions and compared with classical ϑ-SVR. One can see that the proposed method has achieved reduced error rate for most of the kernels when compared with classical ϑ-SVR on PD datasetset indicating that the ϑ-PSVR works better.

The coefficient of determination, R^2 is used to validate the regression efficiency of algorithm. The low value of R^2 represents the weak relationship between the response y and the predictor x and high R^2 represents the strong relationship between the response y and the predictor x. Figure 5.5 depicts the line plot of R^2 values versus number of iterations of proposed ϑ-PSVR and compared with classical ϑ-SVR over linear, RBF, polynomial, sigmoid and Logistic functions. It can be seen that the proposed ϑ-PSVR comparatively achieves higher values of R^2 than ϑ-SVR over almost all kernels indicating that proposed model gives better goodness of fit.

Table 5.1 summarizes the results based on confusion matrix values and performance measures for the proposed ϑ-PSVR and classical ϑ-SVR algorithms with linear, RBF, Polynomial of order 4, , Logistic and sigmoid kernels. Table 5.1 gives the f-score values to asses the generalization

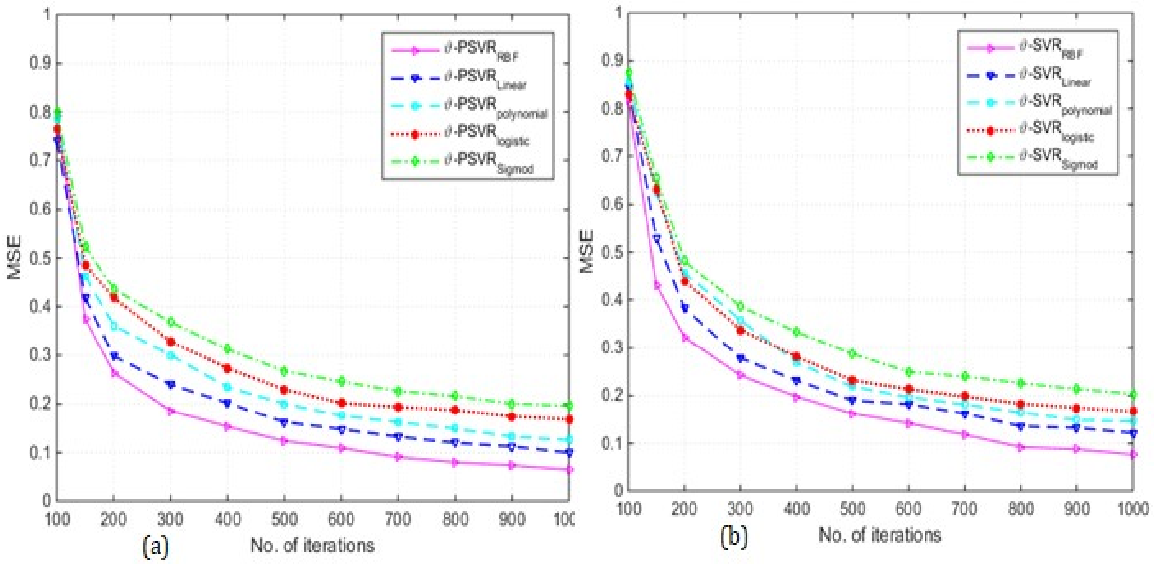


Figure 5.4: Line plots showing the MSE versus Number of iterations of PD dataset evaluated using proposed ϑ-PSVR in Figure 5.4(a) compared with ϑ-SVR in Figure 5.4(b) using Linear, RBF, Ploynomial of order four, Logistic and sigmoid Kernels.

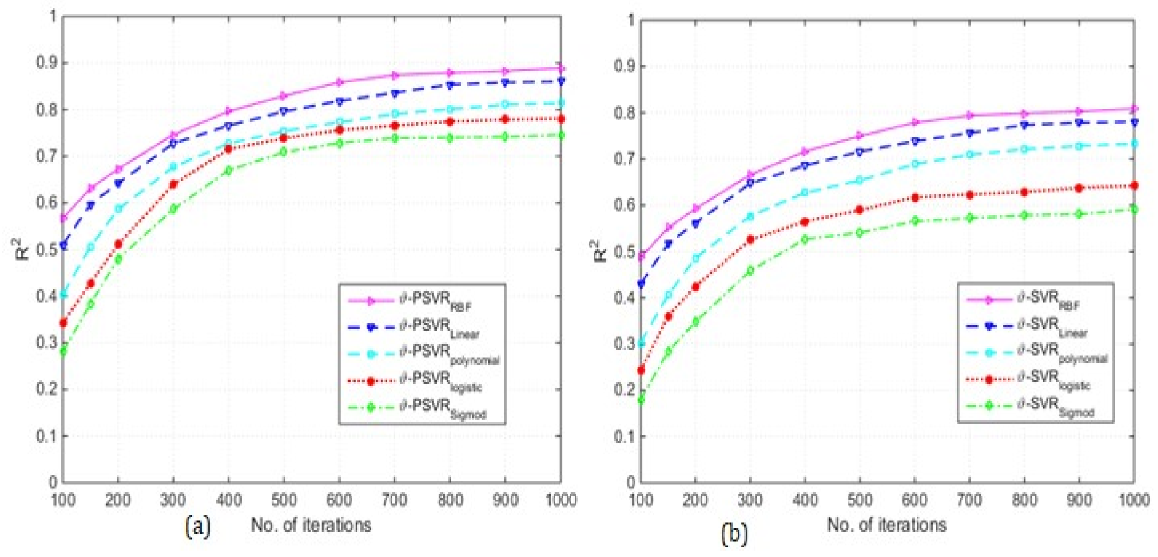


Figure 5.5: Line plots showing the R^2 versus Number of iterations of PD datasetset evaluated using proposed ϑ-PSVR in Figure 5.5(a) compared with ϑ-SVR in Figure 5.5(b) using Linear, RBF, Ploynomial of order four, Logistic and sigmoid Kernels.

performance and the effectiveness of the proposed algorithm which is given by predictive metrices such as MSE and R^2 values. The iteration time of proposed algorithm is represented in CPU time in seconds. The results given in Table 5.1 verified that ϑ-PSVR gives better accuracy with f-score value of 97.74 with MSE=0.131 and R^2=0.758 using RBF kernel. It also costs less time than ϑ-SVR on most of the kernels as given in Table 5.1. Finally, the fitting quality of ε-PSVR is much better and is more competitive when compared with existing techniques.

Table 5.1: Confusion matrix values and performance measures for the ϑ-PSVR and classical ϑ-SVR with different kernels

Kernal	Model	Fscore(%)	CPU time(sec)	MSE	R^2
Linear	ϑ-PSVR	97.71	1.72	0.148	0.741
	ϑ-SVR	97.30	1.98	0.213	0.708
RBF	ϑ-PSVR	97.74	2.26	0.131	0.758
	ϑ-SVR	97.48	2.67	0.198	0.726
Polynomial[4]	ϑ-PSVR	96.80	4.30	0.178	0.724
	ϑ-SVR	95.54	5.40	0.226	0.613
Logistic	ϑ-PSVR	95.90	4.10	0.218	0.698
	ϑ-SVR	94.54	5.20	0.278	0.562
Sigmoid	ϑ-PSVR	97.00	3.50	0.258	0.623
	ϑ-SVR	96.20	3.82	0.292	0.564

The validation of proposed algorithm with RBF kernel is also shown in Table 5.2. The model correctly classified 188 healthy controls out of 195 and 393 early PD patients out of 405, while misclassified 7 healthy controls as early PD group and 12 early PD group as healthy controls. The optimal classification accuracy of 96.73% is achieved using RBF kernel. From above analysis, we can state that the proposed model has improved the performance of PD diagnosis in terms of accuracy and computation time.

Table 5.2: Confusion matrix of ϑ-PSVR with RBF kernel

Observed group	Predicted Group		
	Healthy	PD	% Accuracy
Healthy	188	7	96.40
Early PD	12	393	97.03
Overall %			96.73

The algorithm tries to generate a more smoother regression curve and achieves better prediction accuracy with MSE=0.131 and R^2=0.758 in less computation time of 2.26 sec. compared to classical ϑ-SVR.

5.5 Summary

In this work, we proposed optimized ϑ-PSVR using incremental version of the ϑ-SVR, incorporated with distance weighted discrimination to address the issue of noise in present techniques. The model is simple and robust as it controls the number of support vectors during training. The proposed model is evaluated on Linear and RBF kernels to test the performance against classical ϑ-SVR. The model is validated on MSE, R^2 and computation time against other standard ML techniques. It is shown that the proposed algorithm achieved better performance in less amount of time comparative to its counterparts. Therefore, we conclude that the proposed system has the ability to have great application potential in various ML problems, including complex dataset interpretation.

Chapter 6

Acceleration of incremental learning and decremental unlearning of Support vector machines

The chapter presents a boosting algorithm to improve the performance of incremental learning and decremental unlearning of support vector machines used on imbalanced datasets. In case of imbalanced datasets, the naive SVM learning strategy will not be an effective tool to give satisfactory prediction accuracy since it may be a weak classifier. The proposed boosting algorithm uses asymmetric misclassification cost to modify the training datasets. This modified dataset is used to boost the prediction accuracy of weak SVMs at each training step and a weighted majority vote is used to aggregate the predictions from all of these weak SVM classifiers to get the final class label. Experimental Results on synthetic datasets show that the proposed algorithm enhances the performance of the incremental and decremental SVM.

The main contributions of the proposed work are described as follows:

- A boosting approach based on incremental learning and decremental unlearning is presented to improve the generalization performance of weak non-linear SVM classifiers.
- A boosting dataset of a subsample of original dataset that contains data samples with highest weight values are used to increase the predictive accuracy of weak SVMs.
- The proposed boosting algorithm combines these weak SVMs with asymmetric misclassification cost to modify the training datasets.

- Weak SVMs are then trained by this modified dataset at each iteration.
- The accuracies of these classifiers are then integrated by a weighted majority vote to produce the final class label.
- The proposed method is applied on some synthetic dataset and the algorithm is validated on number of support vectors, error vectors, trajectory of coefficients α_i, trajectory of LOO margin g_i, iterations and training time.

6.1 Introduction

The representation of a time dependent observation x_t at some time points t as tuples (t, x_t), are generally said to be time series data. In time-series prediction ML problems, although training data D is not readily available, however, the samples are usually supplied in a streaming fashion. The task here is to infer a consistent model M_t after every time step based on the present example (x_t, y_t) and the previous model M_{t-1} only [81]. In such scenario, batch SVM algorithms seem computationally not effective as they need to recalculate and retrain a model every time when training data D is updated [82]. Batch SVM algorithm trains input data which is priory available, and can not handle the sheer volume of data which is continuously available in given time stamp. Batch SVM does not continuously integrate new data into already constructed models but instead regularly reconstructs new models from scratch. This is not only very time consuming but also leads to potential intensive computational models. Batch SVM keeps the system weights constant while computing the error associated with each sample in the input and does not scale well enough in accordance with large datasets as they are costly in terms of computational complexity. [22, 23].

Besides, Incremental SVM algorithm is more capable in this case as it uses gradient estimation of weights to train a learning model when a new sample is added to training data D [69, 83, 84].

In addition, the online learning weights are updated on a regular basis using an error computation that is performed for various weights of each input sample. This means that throughout adaptation, the two algorithms explore separate sets of points, but they both converge to the same minimum. It's worth noting that the number of weight adjustments for the same number of data presents differs dramatically between the two strategies. The online method does an update on each sample, while batch does an update on each epoch.

However, the incremental/decremental training of SVM still suffer from computational burden and poor generalization performance mainly for a non-linear SVM classifier when applied to imbalanced datasets or uncertain datasets. Over the years, several techniques were introduced to boost the performance of SVM to solve skewed vector space problem [50, 51, 85]. These boosting techniques are either classifier independent data-driven approaches or classifier based cost-sensitive approaches or combination of both. The recent literature suggests incremental learning and decremental unlearning of SVM by proposing many different approaches to improve overall performance of model [52, 86].

We propose a boosting algorithm to improve the performance of a weak set of non-linear SVM classifiers when used with imbalanced datasets. Weak SVMs are those which underperform when the data set has more noise (target classes are overlapping). Also, when the number of features for each data point exceeds the number of training data samples, it performs poorly. Strong SVMs can achieve a generalization error arbitrarily close to the Bayes error with a sufficiently large training set. Boosting is a way to take several weak SVMs and combine them into a stronger one to eliminate bias, improve model accuracy, and boost performance. Thus the excessive bias encountered due to the skewed vector space problem is handled by a boosted dataset. The proposed algorithm uses a boosted dataset which is a sub sample of the original dataset that are previously misclassified. This boosting algorithm first trains weak SVM classifiers with the notion of incremental learning/decremental unlearning at each iteration using asymmetric misclassification cost. The predictions from all trained algorithms are combined by a weighted majority vote to generate the final prediction. Thus, the final ensemble SVM classifier will have low classification error and also lead to faster training.

The rest of the content of this chapter is organized as follows. Section 6.2 presents the basic equations used to derive the proposed model. Implementation details of the proposed algorithm are described in Section 6.3. Experimental results and validation of the proposed model are discussed in Section 6.4. Chapter is summarised in Section 6.5.

6.2 Preliminaries

SVM classifier of the form $f(x) = w \cdot \phi(x) + b$ learned from the data (x_i, y_j) where $x_i \in \mathbb{R}^m$ and $y_i \in \{+1, -1\} \quad \forall i = 1 \cdots n$ by minimizing the objective function is:

$$\min_{w,b,\xi} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i \quad s.t. \quad y_i(w \cdot \phi(x_i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1 \ldots n. \tag{6.1}$$

This quadratic problem is typically expressed in its dual form after applying Lagrangian function.

$$\min_{\alpha} \quad w = \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\phi(x_i) \cdot \phi(x_j)) - \sum_{i=1}^{n} \alpha_i + b\sum_{i=1}^{n} \alpha_i y_i \quad s.t. \quad 0 \leq \alpha_i \leq C, \tag{6.2}$$

Let Q_{ij} be the positive semi-definitive kernel matrix with Q_{ij} = $y_i y_j K(x_i, x_j)$ where $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$ and the lagrangian α and the bias b, the resulting SVM in dual form is:

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b \tag{6.3}$$

The first-order conditions on w reduce to the KKT conditions, which defines the solution of dual parameters α and b by minimizing the Equation (6.2).

$$g_i = \frac{\partial w}{\partial \alpha_i} = \sum_{j=1}^{n} Q_{ij}\alpha_j + y_i b - 1 = y_i f(x_i) - 1 : \quad \begin{cases} > 0 & \alpha_i = 0 \\ = 0 & 0 \leq \alpha_i \leq C \\ < 0 & \alpha_i = C \end{cases} \tag{6.4}$$

$$h_i = \frac{\partial w}{\partial b} = \sum_{j=1}^{n} y_j \alpha_j = 0 \tag{6.5}$$

SVM initialization and vector migration through learning / unlearning is done by training over individual samples and incrementally/decrementally learns/unlearns by retaining KKT conditions on all previously seen data and adiabatically adds/drops new/old sample to the training vector. KKT conditions partition the training data T and corresponding coefficients α_i, b into three index sets: the set S of margin support vectors strictly on the margin $y_i f(x_i) = 1, g_i = 0$. The set E of error support vectors violating the margin $g_i < 0$ (not necessarily misclassified), and the remaining set of reserve vectors exceeding the margin $g_i > 0$, which is shown in Figure 6.1.

The set E : Error vectors: $E = \{i : |\alpha_i| = C_i\}$

The set S : Support vectors: $S = \{i : 0 < |\alpha_i|< C_i\}$

The set R : Reserve vectors: $R = \{i : |\alpha_i|= 0\}$

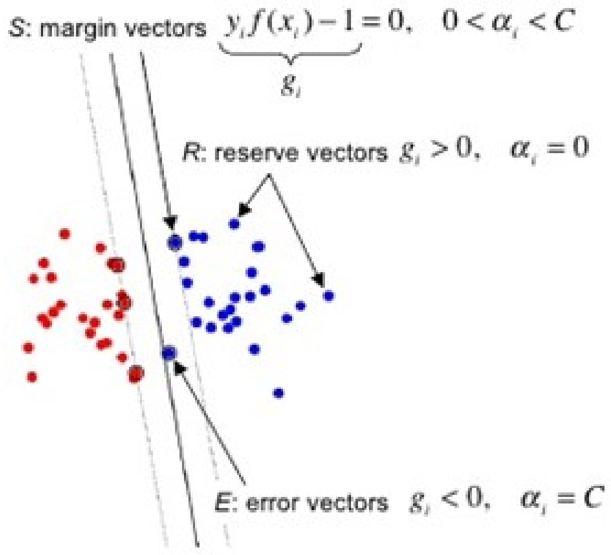

S: margin vectors $\underbrace{y_i f(x_i) - 1}_{g_i} = 0, \quad 0 < \alpha_i < C$
R: reserve vectors $g_i > 0, \quad \alpha_i = 0$
E: error vectors $g_i < 0, \quad \alpha_i = C$

Figure 6.1: Index sets formed by KKT conditions

6.2.1 Incremental/decremental learning of SVM

During the learning phase, algorithm continuously checks for whether a new sample (x_c, y_c) added to the training data T can be inserted to reserve set R as shown in Figure 6.1. If it is not possible to add new sample (x_c, y_c) in R, it tries to add this new sample in either support set S or error set E by maintaining equilibrium of the model on KKT conditions. Each update of the model migrates samples from one index set to another by passing through the support set s such that the **R** matrix (correlated to support vectors) changes accordingly. This method is repeated until all KKT conditions for the samples in T are met.

Decremental unlearning utilizes the concept of Leave-one-out procedure to implement the adiabatic reversal strategy of incremental learning approach. The algorithm unlearns samples decrementally by retaining KKT conditions on all previously used samples and adiabatically forgets old sample from the training vector. Similarly, a list of possible migrations of vectors through the three sets S, E, and R are applied. The solution of $n = n - 1$ is generated by removing a point c (whether it is a margin/error vector) from the training set T. Thus, the solution α_i is given as a function of the coefficients α_i,b and the removed point (x_c, y_c). The solution decides on leaving c out of the given training dataset T that would obtain a misclassification error less than -1. In the decremental unlearning procedure, we assume that the value of α_c approaches to 0; this assumption is only for approximating a classification performance on the training data.

6.3 Proposed Model

The proposed method is a boosting algorithm to enhance the performance of incremental learning and decremental unlearning of support vector machines used on imbalanced datasets. The proposed algorithm first trains a set of weak SVM classifiers with the incremental learning/decremental unlearning strategy at each iteration using gradient estimation of weights with classification error on each input sample. The class probability of these trained weak SVM algorithms are combined by a weighted majority vote to generate the final prediction. Thus the final ensemble SVM classifier will have low classification error and also lead to faster training. further sections details the formulations of proposed model.

6.3.1 Incremental/decremental SVM

The incremental learning procedure is summarised as follows: Let the number of data samples be (n) and a new data sample point be $\{c\}$. If this new data point $\{c\}$ is added to the training set T, then (n) becomes $(n \rightarrow n+1)$, and T^{n+1} becomes $\mathrm{T}^{n} \cup \{c\}$. Thus the given solution $\left\{\alpha_i^{n+1},\ \mathrm{b}^{n+1}\right\}, i = 1, \ldots, n+1$ is given as a function of the present solution $\{\alpha_i^n, \mathrm{b}^n\}$, the present Inverse matrix of R, and the data sample of $(\mathrm{x}_c, \mathrm{y}_c)$. Decremental unlearning is the adiabatic reversal of the incremental learning approach for each data point in the training dataset and is employed by the Leave-one-out procedure The Decremental unlearning procedure is summarised as follows: $(n \rightarrow n-1)$ is generated by removing a point c (whether it is a margin/error vector) from the training set T such that $\mathrm{T} = \mathrm{T}\backslash c$. In this case, the solution $\{\alpha_i \backslash c, \mathrm{b}\backslash c\}$ is given as a function of the coefficients $\{\alpha_i, b\}$, R, and the removed point $(\mathrm{x}_c, \mathrm{y}_c)$. According to Equation (6.5), the solution for $h_c \backslash c$ obtains a misclassification error $(h_c \backslash c)$ less than -1. In the decremental unlearning procedure, we assume that the value of α_c approaches 0; Further details on Incremental and decremental SVM formulations can be referred on Section 3.3.

6.3.2 SVM for class imbalanced data

The modified SVM objective function by using the cost factors and adjusting the cost of false positives and false negatives vectors are defined using regularization parameters. The objective function of SVM therefore can be expressed in the Lagrangian form with two loss functions. Given a set of labeled instances $\{x_i, y_i\}_{i=1}^{n}$, the class prediction function of SVM classifier is formulated in terms of the kernel function K as:

$$f(x) = \operatorname{sign}\left(\sum_{i=1}^{n} y_i \alpha_i K(x, x_i) + b\right) \tag{6.6}$$

where b is the bias and the optimal coefficients are found by maximizing the primal Lagrangian as:

$$\begin{aligned} L_p = \frac{\|w\|^2}{2} + C^+ \sum_{\{i|y_i=+1\}}^{n^+} \xi_i^2 + C^- \sum_{\{j|y_j=-1\}}^{n^-} \xi_j^2 \\ - \sum_{i=1}^{n} \alpha_i \left[y_i (w \cdot x_i + b) - 1 + \xi_i\right] - \sum_{i=1}^{n} \mu_i \xi_i \end{aligned} \tag{6.7}$$

$C^+ \geq \alpha_i \geq 0$, $C^- \geq \alpha_i \geq 0$ $\frac{C^+}{C^-} = \frac{n^-}{n^+}$ and $\mu_i \geq 0$. The points labeled ξ_i^*, $\xi_i^* = \frac{\xi_i}{\|\beta\|}$ are wrong side of the margin. The values of ξ_i^* are calculated as:
$\xi_i = \max\left(0, 1 - y_i \left(\sum_{j=1}^{n} y_j \alpha_j K(x_j, x_i) + b\right)\right)$

6.3.3 Boosting SVM

Incremental/Decremental SVM is boosted by modifying the weights w_i of the training observations x_i in the input space with asymmetric classification error value labelled as ξ^*. Thus, a modified version of the training data which can improve the class prediction is generated. The predictions from weak classifiers are combined now by a weighted majority vote to produce final decision function. The final decision function is:

$$G(x) = sign(\sum_{i=1}^{K} \alpha_k G_k(x)). \tag{6.8}$$

α_k is the weight of the classifier with error rate ε computed by its geometric mean as: $\alpha_k = \lambda \log(1 - \varepsilon)/\varepsilon$, where λ is the empirical parameter to tune the magnitude of the penalty for each iteration. The proposed model with boosted asymmetric classification error and ensemble weighted majority vote is presented in Figure 6.2.

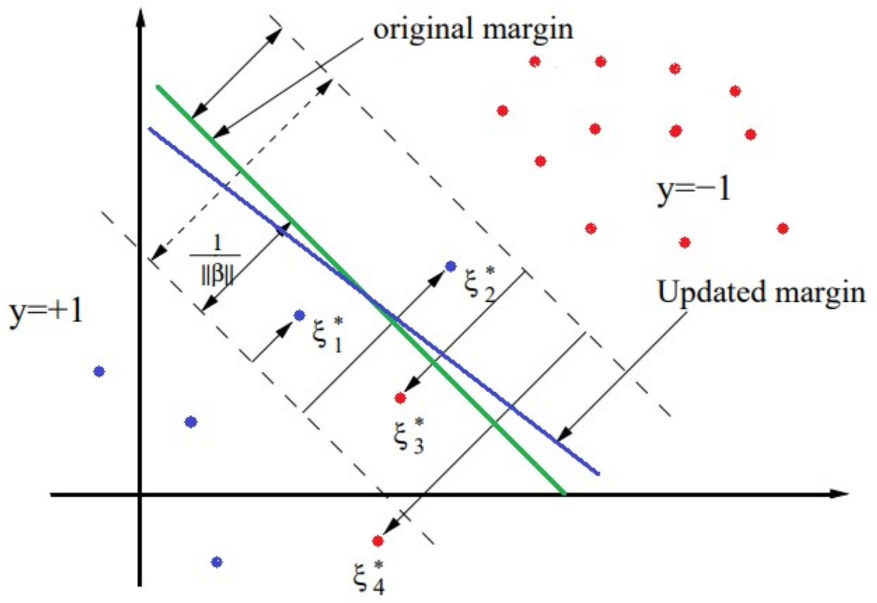


Figure 6.2: SVM with asymmetric misclassification cost

The proposed model generates chunks of classification algorithm and applies the modified data sequentially, thereby producing a sequence of SVM classifiers as: $G_k(x)$, $k = 1, 2, \dots, K$, where K is the numbers of SvMs. Once the predictions from all of the weak classifiers are computed with the modified dataset, they are combined by a weighted majority vote to produce the final prediction $G(x)$ as shown in Equation (6.8). Here the values for $\alpha_1, \alpha_2, \dots, \alpha_K$ are computed by the boosting algorithm and are used to weight the contribution of each respective $G_k(x)$. This strategy will select the most accurate classifier among all.

6.4 Experiments and Results

To evaluate the model, experiments are performed on some linearly separable high-dimensional dataset X of 2-class classification problem generated from Gaussian distribution. We simulate our proposed algorithm on a 3.4 GHz Intel Core i7 2600 CPU with 8 GB RAM using MATLAB 9.2 platform. The parameter, ϵ of the strict restoration adjustment is fixed at 0.1 and C is fixed at 100, respectively in all the experiments.

Figure 6.3 and Figure 6.4 shows the trajectory of the coefficients α_i and g_i and decision surface of the classical SVM using Linear and RBF Kernel for each sample point after adding a new sample in the incremental learning process. The support vectors are denoted by circles. Data-Class1 is denoted by red color and data-class2 is with blue color. Figure 6.3: (a) shows the trajectory of LOO margin g_c as a function of Leave-one-out coefficient α_c. Figure 6.3: (b&c) indicates the incremental learning of coefficients α_i and g_i respectively. Figure 6.3: (d) shows the contour data plot showing the data sequence with the corresponding support vectors(circles) and error vectors (crosses). Figure 6.4: (a) shows the trajectory of LOO margin g_c as a function of Leave-one-out coefficient α_c. Figure 6.4: (b&c) indicates the incremental learning of coefficients α_i and g_i respectively. Figure 6.4: (d) shows the contour data plot showing the data sequence with the corresponding support vectors(circles) and error vectors (crosses).

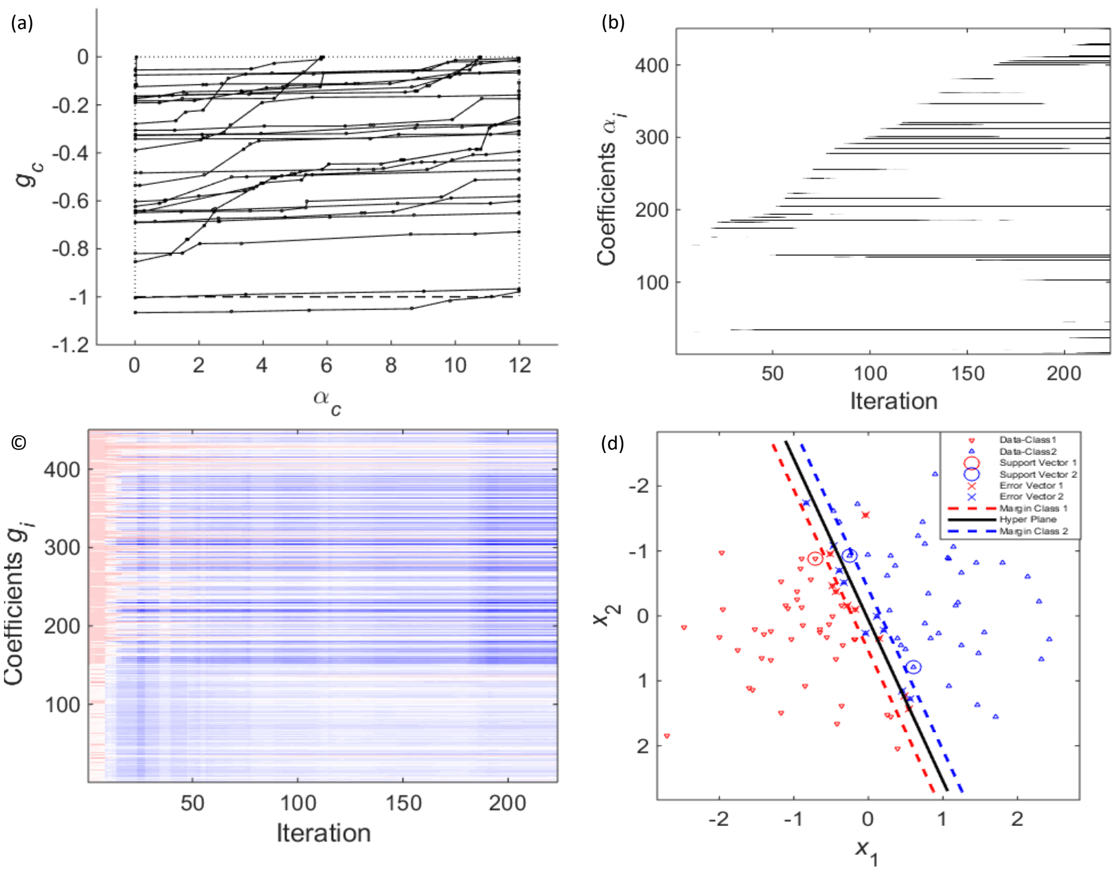


Figure 6.3: Incremental Learning/Decremental unlearning of classical SVM with Linear kernel

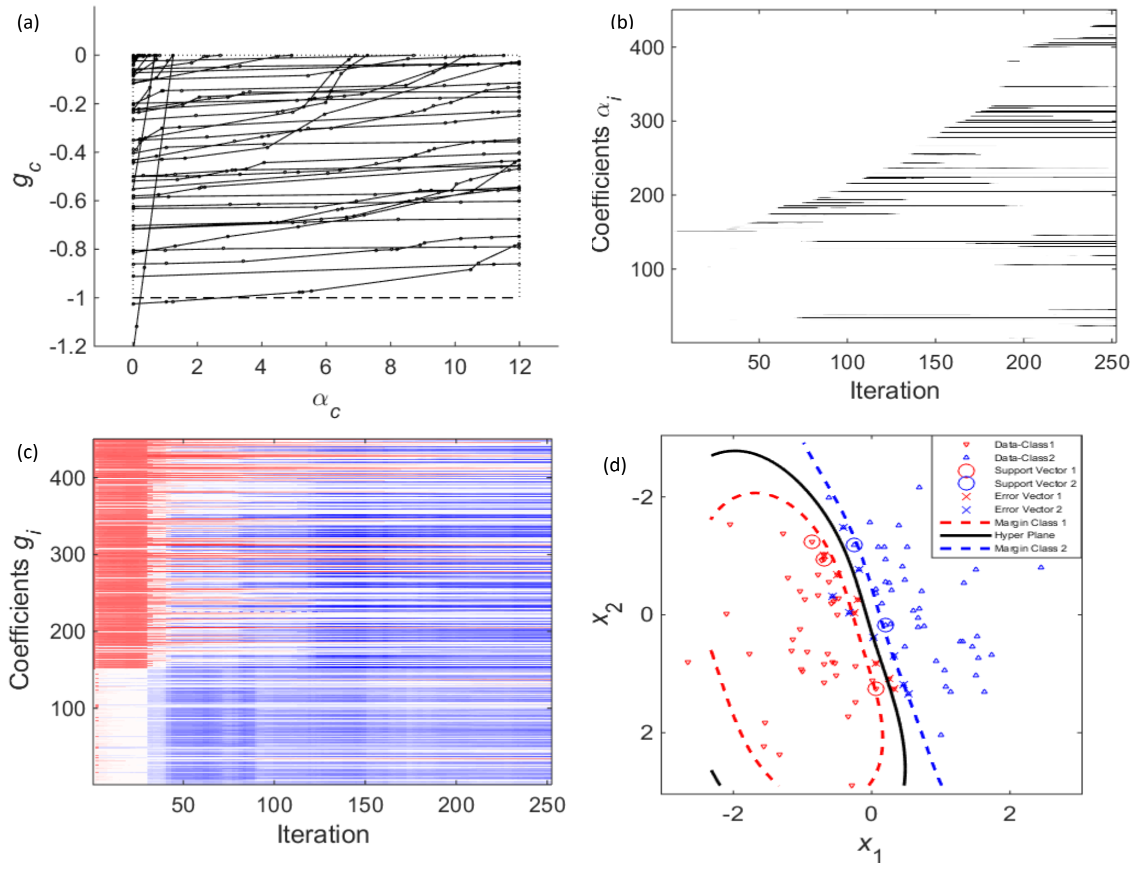


Figure 6.4: Incremental Learning/Decremental unlearning of classical SVM with RBF kernel

Figure 6.5 and Figure 6.6 shows the trajectory of the coefficients α_i and g_i and decision surface of the proposed SVM using Linear and RBF Kernel for each sample point after adding a new sample in the incremental learning process. Figure 6.5 and Figure 6.6: (a-c) gives the incremental learning of the α_c, coefficient α_i and coefficient g_i, respectively and also the iterative updation of the support vectors and error vectors after each iteration for the proposed SVM on Linear and RBF kernels. Figure 6.5 and Figure 6.6: (d) depicts the classification contour plots for incremental learning and decremental unlearning of Proposed SVM over Linear and RBF kernels. In Figure 6.5 and Figure 6.6, the number of error vectors is reduced with faster increments in the $alpha_c$ and g_c values, demonstrating the effectiveness of the proposed SVM algorithm over traditional SVM classifiers. The average values of ten independent runs are computed to produce the results. The results show that learning of the proposed boosted SVM classifier takes less time than learning the standard SVM classifier since the number of iterations is lower with the higher performance.

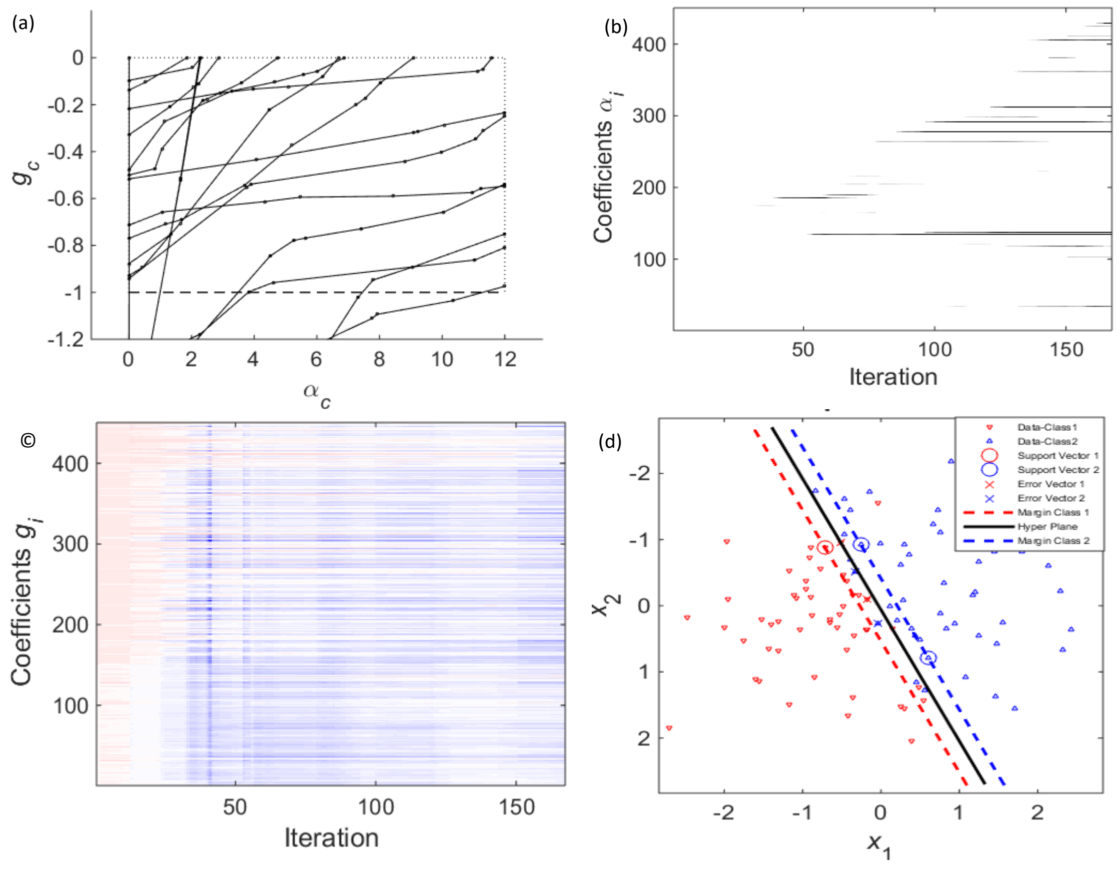


Figure 6.5: Incremental Learning/Decremental unlearning of proposed SVM with Linear kernel

6.5 Summary

This chapter presented a boosting strategy to improve the performance of support vector machines' incremental learning and decremental unlearning on unbalanced datasets. The high-dimensional data, the availability of heterogeneous datasets, particularly for imbalanced datasets, causes the SVM classifier to deteriorate with low convergence and high memory requirements. Boosting is an efficient and simple methodology to enhance the computational and accuracy performance of the weak SVM classifiers. The boosting algorithm proposed here uses a asymetric misclassification cost to define boosted dataset and this boosted dataset is used to train weak SVM classifiers and then the accuracy of these weak SVMs are combined by a weighted majority vote to produce the final classifier. The proposed model is implemented on synthetic datasets and results have shown that the Boosting algorithm speeds up the training time and boosting the performance of the weak SVM classifiers.

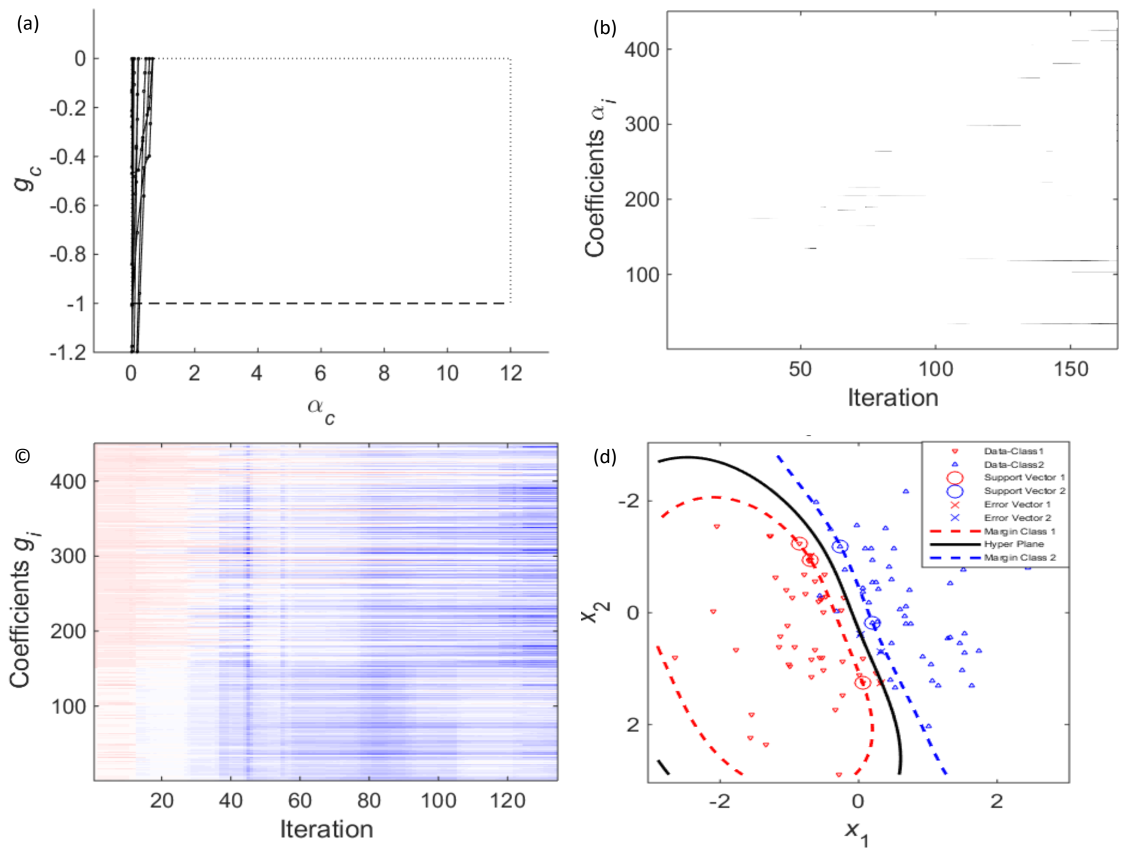


Figure 6.6: Incremental Learning/Decremental unlearning of proposed SVM with RBF kernel

Chapter 7

Conclusion and Future Scope

7.1 Conclusions

This thesis investigates the optimization techniques for most widely used machine learning algorithms called as the Support vector machines for balancing the trade-off between computational complexity and generalization performance.

Chapter 3 describes the fast and robust ensemble SVM which is incrementally trained and converges at faster rates by integrating modified Frank-Wolfe algorithm. The proposed SVM-MFW model allows to learn samples incrementally by maintaining equilibrium with respect to KKT conditions when a new sample is added to the training vector. Convergence rate of proposed algorithm is accelerated by modified Frank-Wolfe method (MFW). The modified FW method uses a new "away step" methodology that can boost the convergence rate of algorithm and can quickly reach acceptable accuracies in the very early iterations. The proposed model is implemented on Parkinson's data taken from PPMI repository and is evaluated on classification accuracy, cross-entropy and computation time using different kernels like linear, polynomial, sigmoid, radial basis function (RBF), Logistic kernel and the results are validated with the classical SVM along with other popular supervised ML algorithms. The accuracy achieved by SVM-MFW using RBF kernel is 98.3%, and prediction accuracy is evaluated using cross-entropy, which is 0.134 and CPU time is 2.32 sec. It can be seen that the proposed model (SVM-MFW) gives better results in comparison to the existing algorithms.

Chapter 4 explains about proposed ε-Support vector regression algorithm that is optimized with the Large margin distribution (LDM) technique by employing modified DCD to enhance the performance. The proposed ε-MDSVR model attempts to make full use of the training set to avoid overfitting and minimizes scattering of the data in ε-tube simultaneously. The proposed method tries to achieve a better generalization performance by optimizing the margin distribution. This margin distribution is characterized by the mean and variance. Proposed algorithm maximizes the mean by considering the distance between the data points nearer to the margin and later attempts to minimize the margin variance. This can be done effectively by employing modified Dual coordinate descent method (DCD). The modified DCD method orderly updates one variable by a single-variable subproblem, where, this variable is selected if it possibly derives the largest decrease in the objective value. This strategy can increase the learning speed and improve the generalization performance. The proposed method is implemented using some popular datasets taken from the UCI machine learning repository. The model is evaluated on Mean square error (MSE) and Regression Coefficient (R^2) using kernels like linear and radial basis function (RBF) and the results are validated with the classical SVR. The proposed algorithm with RBF kernel significantly achieved better prediction accuracy with MSE=0.0074, R^2=0.631 and performs well when compared to the classical SVR and other regression techniques. The proposed algorithm tries to generate more smoother regression curve and achieves better prediction accuracy in less computation time compared to classical SVR.

Chapter 5 details the proposed ϑ-Support vector regression algorithm that is used to handle uncertain data and distribution of data on boundary by applying bounded functions to adjust the number of support vectors and errors. The proposed ϑ-PSVR algorithm addresses the key issues of data piling and overfitting effect due to uncertain data present in classical ϑ-SVR which affects the generalization performance and computational overheads. The proposed method formulates upper bound and lower bound functions on perturbed data. Two special adjustments to the support vectors and penalty parameters are made to enable the model to learn incrementally. The functional margin is optimized by distance weighted discrimination to reduce the distribution of data that defines the regression curve. To evaluate generalization performance, 10-fold cross validation is used. The proposed method is implemented using Parkinson's data taken from PPMI repository. The model is evaluated on Mean square error (MSE), Regression Coefficient (R^2) and computation time using different kernels like linear, polynomial, sigmoid, radial basis function (RBF), Logistic kernel and the results are validated with the classical ϑ-SVR. The algorithm tries to generate a more smoother regression curve and achieves better prediction accuracy with MSE=0.131 and R^2=0.758 in less computation time of 2.26 sec. compared to classical ϑ-SVR.

Chapter 6 presents proposed boosting algorithm to enhance the performance of incremental learning and decremental unlearning of support vector machines used on imbalanced datasets. The proposed boosting algorithm combines the weak SVM classifiers with asymmetric classification error to modify the training data sets. This modified data set is used to train the weak SVMs at each iteration. Later, the predictions from all of these classifiers are then combined by a weighted majority vote to produce the final class label. The proposed method is applied on some synthatic datasets with different sizes and dimensionality. The proposed algorithm is validated on number of support vectors, error vectors, trajectory of coefficients α_i, trajectory of LOO margin g_i, iterations and training time that is compared with classical SVM. The results show that the proposed algorithm boosted the performance of the incremental and decremental SVM.

7.2 Future Scope

- The future work of proposed SVM-MFW can be implemented using the stochastic sub-gradient descent algorithm over some low-sample high dimensional datasets to test the performance.
- The proposed ϑ-PSVR can be applied to solve some real-world data applications to test its generalization.
- A more effective boosting method on our algorithms can tested over different kernel functions to evaluate the performance.

Author's Publications

Journals:

1. Lavanya Madhuri Bollipo and Kadambari KV. "Fast and robust supervised machine learning approach for classification and prediction of parkinson's disease onset".*Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, pages 1-17, 2021 [in press].

2. Lavanya Madhuri Bollipo and Kadambari KV."A novel supervised machine learning algorithm to detect parkinson's disease on its early stages".*Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 12(10), pp. 5257–5276,2021.

3. Lavanya Madhuri Bollipo and Kadambari KV."Optimization of ϑ-Support Vector Regression Algorithm with Distance Weighted Discrimination and Large Margin Distribution". *Journal of Ambient Intelligence & Humanized Computing (AIHC)*. [Under Review]

Patent:

1. Kadambari KV and Bollipo Lavanya Madhuri,"Acceleration of incremental learning and decremental unlearning of Support vector machines". Australia Patent No.: 2021101567

Bibliography

[1] Min Chen, Shiwen Mao, and Yunhao Liu. Big data: A survey. *Mobile networks and applications*, 19(2):171–209, 2014.

[2] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[3] Junfei Qiu, Qihui Wu, Guoru Ding, Yuhua Xu, and Shuo Feng. A survey of machine learning for big data processing. *EURASIP Journal on Advances in Signal Processing*, 2016(1):1–16, 2016.

[4] Alexandra L'heureux, Katarina Grolinger, Hany F Elyamany, and Miriam AM Capretz. Machine learning with big data: Challenges and approaches. *Ieee Access*, 5:7776–7797, 2017.

[5] Omar Y Al-Jarrah, Paul D Yoo, Sami Muhaidat, George K Karagiannidis, and Kamal Taha. Efficient machine learning for big data: A review. *Big Data Research*, 2(3):87–93, 2015.

[6] Roheet Bhatnagar. Machine learning and big data processing: a technological perspective and review. In *International Conference on Advanced Machine Learning Technologies and Applications*, pages 468–478. Springer, 2018.

[7] Sheena Angra and Sachin Ahuja. Machine learning and its applications: A review. In *2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC)*, pages 57–60. IEEE, 2017.

[8] Taiwo Oladipupo Ayodele. Types of machine learning algorithms. *New advances in machine learning*, 3:19–48, 2010.

[9] Amanpreet Singh, Narina Thakur, and Aakanksha Sharma. A review of supervised machine learning algorithms. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 1310–1315. Ieee, 2016.

[10] Pratap Chandra Sen, Mahimarnab Hajra, and Mitadru Ghosh. Supervised classification algorithms in machine learning: A survey and review. In *Emerging technology in modelling and graphics*, pages 99–111. Springer, 2020.

[11] Tammy Jiang, Jaimie L Gradus, and Anthony J Rosellini. Supervised machine learning: a brief primer. *Behavior Therapy*, 51(5):675–687, 2020.

[12] Corinna Cortes and Vladimir Vapnik. Support vector machine. *Machine learning*, 20(3):273–297, 1995.

[13] Lei Wang, Jinhai Sun, and Tuojian Li. Intelligent sports feature recognition system based on texture feature extraction and svm parameter selection. *Journal of Intelligent & Fuzzy Systems*, (Preprint):1–12, 2020.

[14] Lane Maria Rabelo Baccarini, Valceres Vieira Rocha e Silva, Benjamim Rodrigues de Menezes, and Walmir Matos Caminhas. Svm practical industrial application for mechanical faults diagnostic. *Expert Systems with Applications*, 38(6):6980–6984, 2011.

[15] Bernhard Schölkopf, Alex J Smola, Robert C Williamson, and Peter L Bartlett. New support vector algorithms. *Neural computation*, 12(5):1207–1245, 2000.

[16] Bernhard Schölkopf, Alexander J Smola, Francis Bach, et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond.* MIT press, 2002.

[17] Junshui Ma, James Theiler, and Simon Perkins. Accurate on-line support vector regression. *Neural computation*, 15(11):2683–2703, 2003.

[18] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[19] Begüm Demir and Lorenzo Bruzzone. A multiple criteria active learning method for support vector regression. *Pattern recognition*, 47(7):2558–2567, 2014.

[20] Bin Gu, Jian-Dong Wang, Yue-Cheng Yu, Guan-Sheng Zheng, Yu-Fan Huang, and Tao Xu. Accurate on-line ν-support vector learning. *Neural Networks*, 27:51–59, 2012.

[21] Bin Gu, Victor S Sheng, Zhijie Wang, Derek Ho, Said Osman, and Shuo Li. Incremental learning for ν-support vector regression. *Neural Networks*, 67:140–150, 2015.

[22] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

[23] Ziqiao Weng. From conventional machine learning to automl. In *Journal of Physics: Conference Series*, volume 1207, page 012015. IOP Publishing, 2019.

[24] Simon Lacoste-Julien, Martin Jaggi, Mark Schmidt, and Patrick Pletscher. Block-coordinate frank-wolfe optimization for structural svms. In *International Conference on Machine Learning*, pages 53–61. PMLR, 2013.

[25] Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *International Conference on Machine Learning*, pages 427–435. PMLR, 2013.

[26] Ricardo Ñanculef, Emanuele Frandi, Claudio Sartori, and Héctor Allende. A novel frank–wolfe algorithm. analysis and applications to large-scale svm training. *Information Sciences*, 285:66–99, 2014.

[27] Thorsten Joachims. Making large-scale support vector machine learning practical, advances in kernel methods. *Support vector learning*, 1999.

[28] Piyush Kumar, Joseph SB Mitchell, and E Alper Yildirim. Approximate minimum enclosing balls in high dimensions using core-sets. *Journal of Experimental Algorithmics (JEA)*, 8:1–1, 2003.

[29] Ivor W Tsang, James T Kwok, Pak-Ming Cheung, and Nello Cristianini. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6(4), 2005.

[30] Tong Zhang. Sequential greedy approximation for certain convex optimization problems. *IEEE Transactions on Information Theory*, 49(3):682–691, 2003.

[31] E Alper Yildirim. Two algorithms for the minimum enclosing ball problem. *SIAM Journal on Optimization*, 19(3):1368–1391, 2008.

[32] Kenneth L Clarkson. Coresets, sparse greedy approximation, and the frank-wolfe algorithm. *ACM Transactions on Algorithms (TALG)*, 6(4):1–30, 2010.

[33] Martin Jaggi, Simon Lacoste-Julien, Mark Schmidt, and Patrick Pletscher. Block-coordinate frank–wolfe for structural svms. In *NIPS Workshop on Optimization for Machine Learning, Lake Tahoe, NV, USA*, 2012.

[34] Marguerite Frank, Philip Wolfe, et al. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

[35] Emanuele Frandi, Ricardo Nanculef, Maria Grazia Gasparo, Stefano Lodi, and Claudio Sartori. Training support vector machines using frank–wolfe optimization methods. *International Journal of Pattern Recognition and Artificial Intelligence*, 27(03):1360003, 2013.

[36] Bernd Gärtner and Martin Jaggi. Coresets for polytope distance. In *Proceedings of the twenty-fifth annual symposium on Computational geometry*, pages 33–42, 2009.

[37] Teng Zhang and Zhi-Hua Zhou. Large margin distribution machine. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 313–322. ACM, 2014.

[38] Zhi-Hua Zhou. Large margin distribution learning. In *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*, pages 1–11. Springer, 2014.

[39] Zhuang Wang, Koby Crammer, and Slobodan Vucetic. Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale svm training. *The Journal of Machine Learning Research*, 13(1):3103–3131, 2012.

[40] Zongxia Xie and Yingda Li. Large-scale support vector regression with budgeted stochastic gradient descent. *International Journal of Machine Learning and Cybernetics*, 10(6):1529–1541, 2019.

[41] James Stephen Marron, Michael J Todd, and Jeongyoun Ahn. Distance-weighted discrimination. *Journal of the American Statistical Association*, 102(480):1267–1271, 2007.

[42] Xingye Qiao and Lingsong Zhang. Distance-weighted support vector machine. *arXiv preprint arXiv:1310.3003*, 2013.

[43] Yan Wang, Ge Ou, Wei Pang, Lan Huang, and George Macleod Coghill. e-distance weighted support vector regression. *arXiv preprint arXiv:1607.06657*, 2016.

[44] Farid Alizadeh and Donald Goldfarb. Second-order cone programming. *Mathematical programming*, 95(1):3–51, 2003.

[45] Xingye Qiao, Hao Helen Zhang, Yufeng Liu, Michael J Todd, and James Stephen Marron. Weighted distance weighted discrimination and its asymptotic properties. *Journal of the American Statistical Association*, 105(489):401–414, 2010.

[46] Xingye Qiao and Lingsong Zhang. Flexible high-dimensional classification machines and their asymptotic properties. *The Journal of Machine Learning Research*, 16(1):1547–1572, 2015.

[47] Boxiang Wang and Hui Zou. Another look at distance-weighted discrimination. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 80(1):177–198, 2018.

[48] Xin Yee Lam, JS Marron, Defeng Sun, and Kim-Chuan Toh. Fast algorithms for large-scale generalized distance weighted discrimination. *Journal of Computational and Graphical Statistics*, 27(2):368–379, 2018.

[49] Xingye Qiao and Lingsong Zhang. Flexible high-dimensional classification machines and their asymptotic properties. *arXiv preprint arXiv:1310.3004*, 2013.

[50] Benjamin X Wang and Nathalie Japkowicz. Boosting support vector machines for imbalanced data sets. *Knowledge and information systems*, 25(1):1–20, 2010.

[51] R Sundar and M Punniyamoorthy. Performance enhanced boosted svm for imbalanced datasets. *Applied Soft Computing*, 83:105601, 2019.

[52] Honorius Gâlmeanu, Lucian Mircea Sasu, and Razvan Andonie. Incremental and decremental svm for regression. *International Journal of Computers Communications & Control*, 11(6):755–775, 2016.

[53] Bernhard Schölkopf, Chris Burges, and Vladimir Vapnik. Incorporating invariances in support vector learning machines. In *International Conference on Artificial Neural Networks*, pages 47–52. Springer, 1996.

[54] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[55] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

[56] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008.

[57] PS Bradley and OL Mangasarian. Massive data discrimination via linear support vector machines. *Optimization methods and software*, 13(1):1–10, 2000.

[58] Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning. *Advances in neural information processing systems*, pages 409–415, 2001.

[59] Christopher P Diehl and Gert Cauwenberghs. Svm incremental learning, adaptation and optimization. In *Proceedings of the International Joint Conference on Neural Networks, 2003.*, volume 4, pages 2685–2690. IEEE, 2003.

[60] Yangguang Liu, Qinming He, and Qi Chen. Incremental batch learning with support vector machines. In *Fifth World Congress on Intelligent Control and Automation (IEEE Cat. No. 04EX788)*, volume 2, pages 1857–1861. IEEE, 2004.

[61] Alistair Shilton, Marimuthu Palaniswami, Daniel Ralph, and Ah Chung Tsoi. Incremental training of support vector machines. *IEEE transactions on neural networks*, 16(1):114–131, 2005.

[62] Katya Scheinberg, Kristin P Bennett, and Emilio Parrado-Hernández. An efficient implementation of an active set method for svms. *Journal of Machine Learning Research*, 7(10), 2006.

[63] IW-H Tsang, JT-Y Kwok, and Jacek M Zurada. Generalized core vector machines. *IEEE Transactions on Neural Networks*, 17(5):1126–1140, 2006.

[64] Rong-En Fan, Pai-Hsuen Chen, Chih-Jen Lin, and Thorsten Joachims. Working set selection using second order information for training support vector machines. *Journal of machine learning research*, 6(12), 2005.

[65] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[66] Yuh-Jye Lee and Su-Yun Huang. Reduced support vector machines: A statistical theory. *IEEE Transactions on neural networks*, 18(1):1–13, 2007.

[67] Shai Fine and Katya Scheinberg. Efficient svm training using low-rank kernel representations. *Journal of Machine Learning Research*, 2(Dec):243–264, 2001.

[68] Tong Zhang. Sequential greedy approximation for certain convex optimization problems. *IEEE Transactions on Information Theory*, 49(3):682–691, 2006.

[69] Léon Bottou and Yann LeCun. Large scale online learning. *Advances in neural information processing systems*, 16:217–224, 2004.

[70] Hua Ouyang and Alexander Gray. Fast stochastic frank-wolfe algorithms for nonlinear svms. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 245–256. SIAM, 2010.

[71] Peter Bartlett, Yoav Freund, Wee Sun Lee, and Robert E Schapire. Boosting the margin: A new explanation for the effectiveness of voting methods. *The annals of statistics*, 26(5):1651–1686, 1998.

[72] Leo Breiman. Prediction games and arcing algorithms. *Neural computation*, 11(7):1493–1517, 1999.

[73] Lev Reyzin and Robert E Schapire. How boosting the margin can also boost classifier complexity. In *Proceedings of the 23rd international conference on Machine learning*, pages 753–760, 2006.

[74] Liwei Wang, Masashi Sugiyama, Cheng Yang, Zhi-Hua Zhou, and Jufu Feng. On the margin explanation of boosting algorithms. In *COLT*, pages 479–490, 2008.

[75] Wei Gao and Zhi-Hua Zhou. On the doubt about margin explanation of boosting. *Artificial Intelligence*, 203:1–18, 2013.

[76] Mario Martin. On-line support vector machine regression. In *European Conference on Machine Learning*, pages 282–294. Springer, 2002.

[77] Frank Wilcoxon. Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202. Springer, 1992.

[78] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[79] Chao Luo, Chenhao Tan, Xingyuan Wang, and Yuanjie Zheng. An evolving recurrent interval type-2 intuitionistic fuzzy neural network for online learning and time series prediction. *Applied Soft Computing*, 78:150–163, 2019.

[80] Dongdong Zhang, Wenguo Xiang, Qiwei Cao, and Shiyi Chen. Application of incremental support vector regression based on optimal training subset and improved particle swarm optimization algorithm in real-time sensor fault diagnosis. *Applied Intelligence*, pages 1–16, 2020.

[81] Chi-Jie Lu, Tian-Shyug Lee, and Chih-Chou Chiu. Financial time series forecasting using independent component analysis and support vector regression. *Decision Support Systems*, 47(2):115–125, 2009.

[82] Dun Liu, Tianrui Li, and Decui Liang. Incorporating logistic regression to decision-theoretic rough sets for classifications. *International Journal of Approximate Reasoning*, 55(1):197–210, 2014.

[83] Pavel Laskov, Christian Gehl, Stefan Krüger, and Klaus-Robert Müller. Incremental support vector learning: Analysis, implementation and applications. *Journal of machine learning research*, 7(Sep):1909–1936, 2006.

[84] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

[85] Yang Liu, Aijun An, and Xiangji Huang. Boosting prediction accuracy on imbalanced datasets with svm ensembles. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 107–118. Springer, 2006.

[86] Rasha Kashef. A boosted svm classifier trained by incremental learning and decremental unlearning approach. *Expert Systems with Applications*, page 114154, 2020.