Lagrangian Twin Support Vector Regression Based on Twin Support Vector Machine (LTWSVR)

A dissertation submitted to the Jawaharlal Nehru University
in partial fulfillment of the requirements
for the award of the degree of

MASTER OF TECHNOLOGY

IN

COMPUTER SCIENCE AND TECHNOLOGY

BY

SUBHASH CHANDRA PRASAD

SCHOOL OF COMPUTER AND SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI – 110067
INDIA

JULY 2015

SCHOOL OF COMPUTER AND SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI – 110067

DECLARATION

This is to certify that the dissertation entitled "**Lagrangian Twin Support Vector Regression Based on Twin Support Vector Machine (LTWSVR)**" is being submitted to the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, in partial fulfillment of the requirements for the award of the degree of **Master of Technology in Computer Science & Technology,** is a record of bonafide work carried out by me under the supervision of **Prof. S. Balasundaram**.

The matter embodied in the dissertation has not been submitted in part or full to any University or Institution for the award of any degree or diploma.

SSprasad

Subhash Chandra Prasad
(Student)

Enrollment No.13/10/MT/28

Date: 04/07/2015

SCHOOL OF COMPUTER AND SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI – 110067

CERTIFICATE

This is to certify that this dissertation entitled **"Lagrangian Twin Support Vector Regression Based on Twin Support Vector Machine (LTWSVR)"** being submitted by Mr. **Subhash Chandra Prasad** to the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, for the award of degree of **Master of Technology** in **Computer Science & Technology**, is a research work carried out by him under the supervision of **Prof. S. Balasundaram.**

[illegible]	[illegible]
Prof. S. Balasundaram	Prof. C.P. Katti
(Supervisor)	(Dean)
SC&SS, JNU, New Delhi	SC&SS, JNU, New Delhi
Date: 05/07/15	Date: Dean School of Computer & Systems Sciences Jawaharlal Nehru University New Delhi-110067

DeDicateD to

MY PARENTS & GOD, who gave me all the beautiful things I needed.

ACKNOWLEDGEMENTS

I take this opportunity to express a deep sense of gratitude for my supervisor Prof. S. Balasundaram for his outstanding guidance, support and care throughout the year while I was working on my dissertation. Successful completion of the present research effort not has been possible without him, who not only served as my guide but also encouraged and challenged me throughout my research.

I wish to thank Professor C. P. Katti, Dean, School of Computer and Systems Sciences, Jawaharlal Nehru University, for providing the necessary research facilities.

I would like to express my sincere thanks to Dr. T.V. Vijay Kumar who has been very kind and generous in giving his precious time for personal and academic discussions.

I am thankful to my colleagues Deepak Gupta, Gagandeep, Yogendra Meena and Tarundeep Kaur Saini for their help and support. I would also like to thank all my friends for the great time I have always had with them. Apart from the fun, some of my friends have also been a great source of learning and inspiration. Thanks a lot to one and all and special thanks to Saurabh, Vikas and Ram Nayan Verma.

Finally, I must admit that a “thank you” is something too small for the role played by my parents and my supervisors in shaping this dissertation. In their absence, this desertation would have been very different or perhaps would not have existed at all.

Subhash Chandra Prasad
JNU, New Delhi

ABSTRACT

In this work, a simple reformulation of the Lagrangian dual of the 2-norm twin support vector machine based regression (TWSVR) is proposed as unconstrained minimization problem. The proposed Lagrangian twin support vector regression based on twin support vector machine (LTWSVR) requires at the outset the inverse of a matrix but this can be expressed as matrix subtraction of identity matrix by a scalar multiple of the inverse of a positive semi-definite matrix. The LTWSVR is solvable by computing the zeros of its gradient. Further it is proposed to solve this problem by simple iterative methods: functional iterative method (FLTWSVR), Newton method (NLTWSVR) and Generalized derivative approach (GLTWSVR). To demonstrate the effectiveness of LTWSVR, numerical experiments were performed on a number of interesting synthetic and real-world benchmark datasets. The results obtained show similar or better generalization performance with much faster learning speed in comparison with SVR, TSVR and TWSVR.

Contents

List of Figures

List of Tables

List of Symbols

Symbol	
R	Real numbers
R_+	Non negative real numbers
$\mathbf{x} \in R^n$	Input vector
$\bar{y}$	Output vector
$\mathbf{x}^t$	Transpose of $\mathbf{x}$
$S = \{(\mathbf{x}_i, y_i)\}, i = 1, \ldots, m$	Training set
$\tilde{\mathbf{x}}_i^t \tilde{\mathbf{x}}_j$	Inner product between $\mathbf{x}_i$ and $\mathbf{x}_j$
$\phi(\mathbf{x})$	Mapping from input space into Feature space via the function ϕ
$\mathbf{w}$	Weight vector in R^n
b	Threshold
$k(\mathbf{x}_i, \mathbf{x}_j)$	Kernel function
K	Kernel matrix
$\lVert \cdot \rVert_p$	p -norm
$\lVert \cdot \rVert$	2 -norm
C	Penalty parameter
$\boldsymbol{\xi}$	Vector of slack variables
$\mathbf{u}$	Vector of dual variables, Lagrange multiplier vector
$\mathbf{x}_+$	Vector $\mathbf{x}$ with all negative components set to zero
$(\mathbf{x}_*)_i$	$\begin{cases} 1 & \text{if } x_i > 0 \\ 0.5 & \text{if } x_i = 0, \quad \text{where } (.)_i \text{ represents } i^{th} \text{ component} \\ 0 & \text{if } x_i < 0 \end{cases}$
$\mathbf{x} \geq \mathbf{0}$	Each component of $\mathbf{x}$ is nonnegative
$diag(\mathbf{x})$	A diagonal matrix having components of $\mathbf{x}$ as its diagonal elements
$\nabla f(\mathbf{x}) = \left(\partial f / \partial x_1, \ldots, \partial f / \partial x_n\right)^t$	Gradient of function f
$\nabla^2 f(\mathbf{x}) = \left(\partial^2 f / \partial x_i \partial x_j\right)^t$	Hessian matrix

List of Abbreviations

Abbreviation	
ANN	Artificial Neural Network
CQPP	Constrained Quadratic Programming Problem
ERM	Empirical Risk Minimization
GEPSVM	Generalized Eigenvalues and Proximal Support Vector Machine
LTWSVR	Lagrangian Twin Support Vector Regression Based on Twin Support Vector Machine
SRM	Structural Risk Minimization
SVM	Support Vector Machine
SVR	Support Vector Regression
TSVM	Twin Support Vector Machine
TSVR	Twin Support Vector regression
TWSVR	Twin Support Vector Machine Based Regression

Chapter 1

Introduction: Support Vector Machine in Regression

1.1 General

Support Vector Machines (SVMs) also known as Kernel Machines are one of the best supervised learning technique for both classification problems as pattern recognition and regression problems as function approximation, proposed by Russian Scientist Vladimir Naumovich Vapnik (Vapnik, 2000). They embody several features from statistical learning theory, machine learning, and optimization theory, and employ kernel functions as one of their essential ingredients. SVM has been the most promising machine learning method due to its formulation based on the novel paradigm vested in the structural risk minimization induction principle (SRM principal) (Cristianini and Shawe-Taylor, 2000; Vapnik, 2000; Kecman, 2001) and can effectively avoid the local minimum and overfitting problem in classical machine learning methods such as neural networks (NNs), which performs Empirical Risk Minimization (ERM). Unlike ERM which minimizes training error, structural risk minimization (SRM) minimizes the upper bound on expected risk or generalization error consists of both an empirical risk term and regularization term that measures the complexity of the machine (norm of the classifier or regressor) and is superior than ERM (Gunn, 1998). This is the difference that powers the SVM to have good generalization i.e. better prediction on previously unseen data (Burges & Scholkopf , 2007; Kecman, 2001).

Mathematically, classification and regression problems in SVM have been shown as optimization problems having quadratic objective function and linear constraints; i.e. they are convex programming problems with unique solution (Cristianini & Shawe Taylor, 2000; Kaufman, 1999; Vapnik, 2000). A clear benefit of SVM is that its solution is sparse; i.e. only some of the samples contribute in determination of the decision function (Gunn, 1998; Scholkopf & Smola, 2002).

SVM has been successfully applied to real world data analysis problems in many fields (bioinformatics, handwriting recognition, stock market etc), often providing better results than (or comparable outcomes to) ANNs (Kecman, 2001). In comparison with most other learning techniques, SVMs show improved result in pattern recognition and regression estimations problems of practical importance such as: Combustion engine detection (Rychetsky et al.,1999), Face detection (Osuna, Freund & Cirosi, 1997), Financial time series forecasting (Mukherjee, Osuna, & Girosi, 1997; Tay & Cao, 2001, Kim, 2003), Handwritten digit recognition (Burges & Scholkopf, 1997; Cortes & Vapnik, 1995), Object recognition (C. P. Papageorgiou, M. Oren, & T. Poggio, 1998), Marketing (Ben-David & Lindenbaum, 1997), medical diagnosis (Tarassenko et al.,1995), text categorization (Joachims, 2002), estimating manufacturing yields (Stoneking, 1999) etc.

1.2 The Regression Problem

Assume that we are given a training data set of m samples

$$S = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_m, y_m)\}, \quad \mathbf{x}_i = (x_{i1}, \cdots, x_{in})^t \in X \subseteq R^n, y_i \in Y \subseteq R. \tag{1.1}$$

with $\mathbf{x}_1, \cdots, \mathbf{x}_m$ drawn according to an unknown probability distribution $P(\mathbf{x}, y)$ and $y_i = f_{true}(\mathbf{x}_i)$ for all $i \in [1, m]$. Let H be a hypothesis set of linear functions mapping *X* to *Y*, i.e.

$$H = \{ f \mid f(\mathbf{x}) = \mathbf{w}^t \mathbf{x} + b\}, \qquad \mathbf{w}, \mathbf{x} \in R^n, \quad b \in R \tag{1.2}$$

We denote the loss function by $L: Y \times Y \to R_+$ used to measure the magnitude of error. The most commonly used loss function in regression is the *quadratic loss* L_2 defined as $L(y, y' = f(\mathbf{x})) = |y - y'|^2 \quad \forall y, y' \in Y$, where y and y' are actual and predicted output values corresponding to a given input $\mathbf{x}$ or being a more general L_p loss defined by $L(y, y' = f(\mathbf{x})) = |y - y'|^p \quad \forall y, y' \in Y$ and for some $p \geq 1$.

The task of regression is to find a hypothesis $f \in H$ that minimizes the expected risk or generalization error (Kecman, 2001),

$$R(h) = \underset{\mathbf{x} \sim P(\mathbf{x}, y)}{\mathrm{E}} \{L(y, f(\mathbf{x}))\} = \int L(y, f(\mathbf{x})) dP(\mathbf{x}, y), \tag{1.3}$$

with respect to target *f* based on the training data set *S*.

Throughout in this work, we denote the training dataset inputs $\{\mathbf{x}_i\}_{i=1,\cdots,m}$ by a matrix $A \in R^{m\times n}$ whose i^{th} row $A_i \in R^n$ represents the i^{th} training sample and their corresponding outputs $\{y_i\}_{i=1,\cdots,m}$ by an output vector $\mathbf{y} = (y_1, \cdots, y_m)^t$ respectively.

1.3 Support Vector Regression

SVM can be successfully applied in regression i.e. function approximation problem by the introduction of a novel loss (error) function different from the classical quadratic error function. This is the $\varepsilon-$insensitive loss function for support vector regression (SVR) proposed by Vapnik (Vapnik, 2000).

In SVR, the linear regressor or linear hyperplane (approximation function $f : R^n \to R$) for regression problem given in section 1.2 can be defined as

$$f(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + b \tag{1.4}$$

and can be obtained by simultaneously minimizing the weight vector norm $(\mathbf{w})$ and empirical risk which can be written as an unconstrained optimization problem,

$$\min \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} L(y_i, f(\mathbf{x}_i)) \tag{1.5}$$

where $C > 0$ is the regularization parameter and $L(\cdot,\cdot)$ is the error loss function. For ε-insensitive error loss function, $L(y_i, f(\mathbf{x}_i)) = |y_i - f(\mathbf{x}_i)|_\varepsilon$, the problem (1.5) becomes

$$\min_{\mathbf{w},b\in R^{n+1}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} |y_i - (A_i\mathbf{w} + b)|_\varepsilon \tag{1.6}$$

Error Loss Function	**Function definition** $L(y, f(\mathbf{x}))$
Laplacian or Linear loss	$\|y - f(\mathbf{x})\|$
Gaussian or Quadratic loss	$\|y - f(\mathbf{x})\|^2$
Huber's Robust loss (Gunn, 1998)	$\begin{cases} \frac{1}{2}(y - f(\mathbf{x}))^2 & if \|y - f(\mathbf{x})\| \le \varepsilon \\ \varepsilon\|y - f(\mathbf{x})\| - \frac{\varepsilon^2}{2} & \text{otherwise} \end{cases}$
Linear $\varepsilon-$ insensitive loss (Vapnik, 2000)	$\begin{cases} 0 & \|y - f(\mathbf{x})\| \le \varepsilon \\ \|y - f(\mathbf{x})\| - \varepsilon, & \text{otherwise} \end{cases}$
Quadratic $\varepsilon-$ insensitive loss	$\max\{0, \|y - f(\mathbf{x})\| - \varepsilon\}^2$

Table 1.1: Common Loss functions

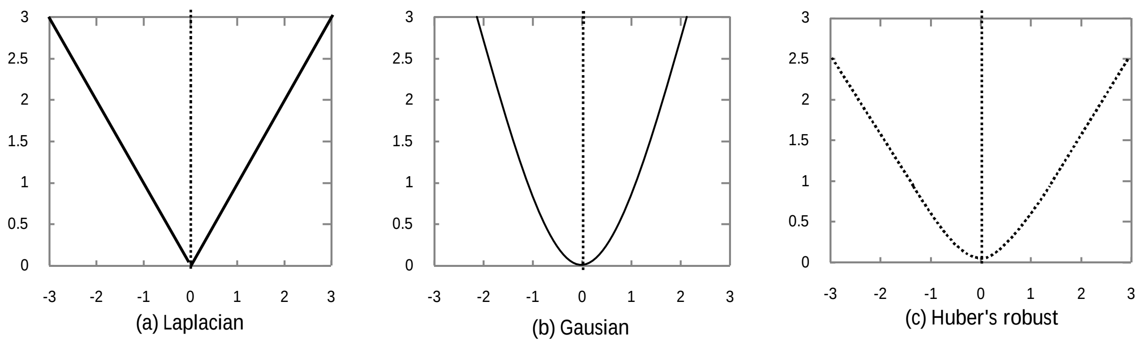


Figure 1.1: Graphs of Loss Functions: Laplacian, Gaussian and Huber’s robust

The loss functions summarized in Table 1.1 can be used in derivation of support vector algorithms that lead to quadratic programming problems but only linear and quadratic $\varepsilon-$insensitive loss functions will produce sparse representation of the regressor i.e. approximation function.

The Laplacian (Linear or absolute) loss function in Figure 1.1(a) corresponds to the median of the conditional distribution and its optimization means predicting the (conditional) median of data. Gaussian (Quadratic) loss function in Figure 1.1(b) like traditional least square method penalizes the large deviation from target outputs while ignoring the small residuals and its optimization means predicting the (conditional) mean of the data. Laplacian loss function is less sensitive to outliers than Gaussian loss function. Huber loss functions in Figure 1.1(c) is a robust loss function where nothing specific is known about the distribution describing the data.

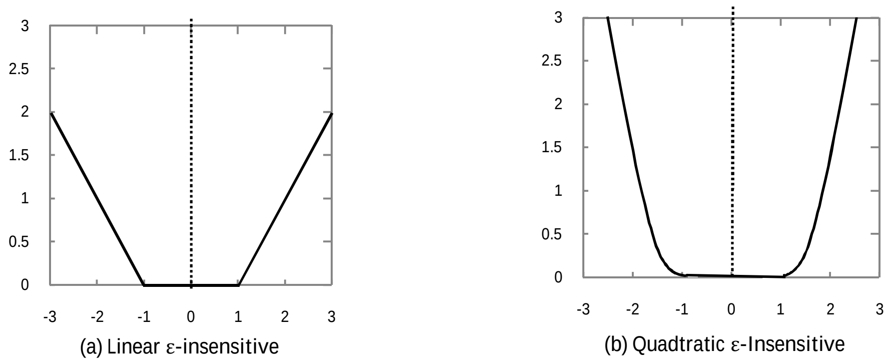


Figure 1.2: Graphs of Loss functions: Linear and Quadratic ε- insensitive

The linear and quadratic ε-insensitive loss functions in Figure 1.2 can be seen as the generalizations of the Laplacian and Gaussian loss functions.

1.4 Linear SVR

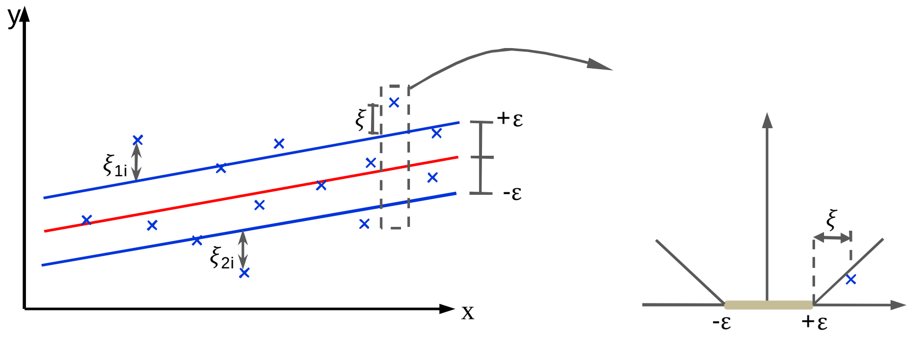


Figure 1.3: Linear Support Vector Regression with ε- insensitive loss

The objective of linear SVR is to find a function $f(\mathbf{x})$ (1.4) that comes closest to training data (1.1) but for all training data of at most $\varepsilon-$deviation from their corresponding targets y_i is allowed and the function must be made as flat as possible.

In Figure 1.3, the deviation of data points (denoted by $\times$ symbol) are captured by introducing vectors of slack variables i.e. ξ_1, ξ_2 .Only data points outside the ε-tube are considered as training errors. Vectors ξ_1 and ξ_2 measure the deviations of data points that are above and below the ε-tube respectively.

1.4.1 SVR with Linear ε-Insensitive Loss

Using the ε-insensitive loss function, the optimization problem (1.5) for regression problem described in section 1.3 can be written as a constrained optimization problem i.e.

$$\min_{\mathbf{w}, b \in R^{n+1}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}(\xi_{1i}+\xi_{2i}) \tag{1.7}$$

subject to
$$y_i - (\mathbf{w}^t\mathbf{x}_i + b) \le \varepsilon + \xi_{1i},$$
$$(\mathbf{w}^t\mathbf{x}_i + b) - y_i \le \varepsilon + \xi_{2i},$$
$$\xi_{1i}, \xi_{2i} \ge 0, \quad i = 1,\ldots,m$$

Optimization problem (1.7) can also be written as

$$\min_{(\mathbf{w},b,\xi_1,\xi_2)\in R^{n+1+m+m}} \quad \frac{1}{2}\mathbf{w}^t\mathbf{w} + C(\mathbf{e}^t\xi_1 + \mathbf{e}^t\xi_2)$$

subject to

$$\begin{aligned} &\mathbf{y} - A\mathbf{w} - b\mathbf{e} \le \varepsilon\mathbf{e} + \xi_1, \\ &A\mathbf{w} + b\mathbf{e} - \mathbf{y} \le \varepsilon\mathbf{e} + \xi_2, \\ &\xi_{1i}, \xi_{2i} \ge 0 \quad i = 1,...,m \end{aligned} \tag{1.8}$$

where $\xi_1 = (\xi_{11},\cdots,\xi_{1m})^t, \xi_2 = (\xi_{21},\cdots,\xi_{2m})^t$ are vectors of slack variables and $\mathbf{e} = (1,\cdots,1)^t \in R^m$ is the vector of ones. The constant $C > 0$ influences the trade-off between the flatness of *f* and the amount up to which deviation larger then ε are tolerated. An increase in value of *C* penalizes the large errors while decrease in value penalizes small errors.

Introducing Lagrange multipliers $\mathbf{u}_1 = (u_{11},\cdots,u_{1m})^t, \mathbf{u}_2 = (u_{21},\cdots,u_{2m})^t$ the Lagrangian function in primal variables of the above problem (1.8) can be formed as

$$\begin{aligned} L(\mathbf{w},b,\xi_1,\xi_2,\mathbf{u}_1,\mathbf{u}_2) = \frac{1}{2}\mathbf{w}^t\mathbf{w} + C(\mathbf{e}^t\xi_1 + \mathbf{e}^t\xi_2) - (\varepsilon\mathbf{e}^t\mathbf{u}_1 + \xi_1^t\mathbf{u}_1 - \mathbf{y}^t\mathbf{u}_1 + (A\mathbf{w})^t\mathbf{u}_1 + b\mathbf{e}^t\mathbf{u}_1) \\ -(\varepsilon\mathbf{e}^t\mathbf{u}_2 + \mathbf{e}^t\xi_2^t\mathbf{u}_2 + \mathbf{y}^t\mathbf{u}_2 - (A\mathbf{w})^t\mathbf{u}_2 - b\mathbf{e}^t\mathbf{u}_2) \end{aligned} \tag{1.9}$$

According to KKT conditions (Karush, 1939; Kuhn et al., 1951) the partial derivative of (1.9) with respect to the primal variables $\mathbf{w},b,\xi_1,\xi_2$ vanish at optimality

i.e.

$$\begin{aligned} &\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - A^t(\mathbf{u}_1 - \mathbf{u}_2) = 0 \quad \Rightarrow \mathbf{w} = A^t(\mathbf{u}_1 - \mathbf{u}_2) \\ &\frac{\partial L}{\partial b} = e^t(\mathbf{u}_1 - \mathbf{u}_2), \quad \frac{\partial L}{\partial \xi_1} = C\mathbf{e} - \mathbf{u}_1 = 0, \quad \frac{\partial L}{\partial \xi_2} = C\mathbf{e} - \mathbf{u}_2 = 0 \end{aligned}$$

Substituting these results back into (1.9), the dual of problem (1.8) can be formed as,

$$\min_{(\mathbf{u}_1,\mathbf{u}_2)b\in R^{m+m}} \quad \frac{1}{2}(\mathbf{u}_1 - \mathbf{u}_2)^t AA^t(\mathbf{u}_1 - \mathbf{u}_2) - \mathbf{y}^t(\mathbf{u}_1 - \mathbf{u}_2) + \varepsilon e^t(\mathbf{u}_1 + \mathbf{u}_2)$$

subject to $\quad \mathbf{e}^t(\mathbf{u}_1 - \mathbf{u}_2) = 0$ and $\mathbf{0} \le \mathbf{u}_1, \mathbf{u}_2 \le C\mathbf{e}$ (1.10)

Now, for any example $\mathbf{x} \in R^n$, the regressor function (1.4) can predict its output as

$$f(\mathbf{x}) = (\mathbf{u}_1 - \mathbf{u}_2)^t A\mathbf{x} + b. \tag{1.11}$$

1.4.2 SVR with Quadratic ε -Insensitive Loss

Using the quadratic ε -insensitive loss function, the optimization problem (1.5) for regression problem (1.2) can be written as a constrained optimization problem i.e.

$$\min_{(\mathbf{w},b,\boldsymbol{\xi}_1,\boldsymbol{\xi}_2)\in R^{n+1+m+m}} \ \frac{1}{2}\mathbf{w}^t\mathbf{w}+\frac{C}{2}(\boldsymbol{\xi}_1^t\boldsymbol{\xi}_1+\boldsymbol{\xi}_2^t\boldsymbol{\xi}_2)$$

subject to

$$\begin{aligned}&\mathbf{y}-A\mathbf{w}-b\mathbf{e}\le\varepsilon\mathbf{e}+\boldsymbol{\xi}_1,\\&A\mathbf{w}+b\mathbf{e}-\mathbf{y}\le\varepsilon\mathbf{e}+\boldsymbol{\xi}_2,\\&\xi_{1i},\xi_{2i}\ge 0\quad i=1,...,m\end{aligned}\tag{1.12}$$

where $\boldsymbol{\xi}_1=(\xi_{11},\cdots,\xi_{1m})^t$ and $\boldsymbol{\xi}_2=(\xi_{21},\cdots,\xi_{2m})^t$ are vectors of slack variables and $C>0$. This formulation has only $2m$ non-negative and linear constraints.

Introducing Lagrange multipliers $\mathbf{u}_1=(u_{11},\cdots,u_{1m})^t,\mathbf{u}_2=(u_{21},\cdots,u_{2m})^t$ the Lagrangian function in primal variables of problem (1.12) can be formed as

$$\begin{aligned}L(\mathbf{w},b,\boldsymbol{\xi}_1,\boldsymbol{\xi}_2,\mathbf{u}_1,\mathbf{u}_2)=&\frac{1}{2}\mathbf{w}^t\mathbf{w}+\frac{C}{2}(\boldsymbol{\xi}_1^t\boldsymbol{\xi}_1+\boldsymbol{\xi}_2^t\boldsymbol{\xi}_2)-(\varepsilon\mathbf{e}^t\mathbf{u}_1+\boldsymbol{\xi}_1^t\mathbf{u}_1-\mathbf{y}^t\mathbf{u}_1+(A\mathbf{w})^t\mathbf{u}_1+b\mathbf{e}^t\mathbf{u}_1)\\&-(\varepsilon\mathbf{e}^t\mathbf{u}_2+\mathbf{e}^t\boldsymbol{\xi}_2^t\mathbf{u}_2+\mathbf{y}^t\mathbf{u}_2-(A\mathbf{w})^t\mathbf{u}_2-b\mathbf{e}^t\mathbf{u}_2)\end{aligned}\tag{1.13}$$

Proceeding as in previous section, the dual of (1.13) can be obtained in the following form,

$$\min_{(\mathbf{u}_1,u_2)\in R^{m+m}} \ \frac{1}{2}(\mathbf{u}_1-\mathbf{u}_2)^t(AA^t+\frac{I}{C})(\mathbf{u}_1-\mathbf{u}_2)-\mathbf{y}^t(\mathbf{u}_1-\mathbf{u}_2)+\varepsilon e^t(\mathbf{u}_1+\mathbf{u}_2)$$

subject to

$$\mathbf{0}\le\mathbf{u}_1,\mathbf{u}_2$$

where I is an identity matrix of size *m*. The term $1/C$ is added to diagonal of Hessian matrix, which ensures positive definiteness of Hessian and stabilizes the solution and there is no upper bound on $\mathbf{u}_1,\mathbf{u}_2$.

For any input $\mathbf{x}\in R^n$, the regressor function (1.4) becomes

$$f(\mathbf{x})=(\mathbf{u}_1-\mathbf{u}_2)^tA\mathbf{x}+b\tag{1.14}$$

1.5 Nonlinear SVR

The practical application of support vector regression procedure is only possible with linear functions because we only have an optimality criterion for linear functions (linear hyperplanes). There are no general results for nonlinear functions. For many applications, a linear solution does not provide good performances; so at many times a nonlinear approach is needed.

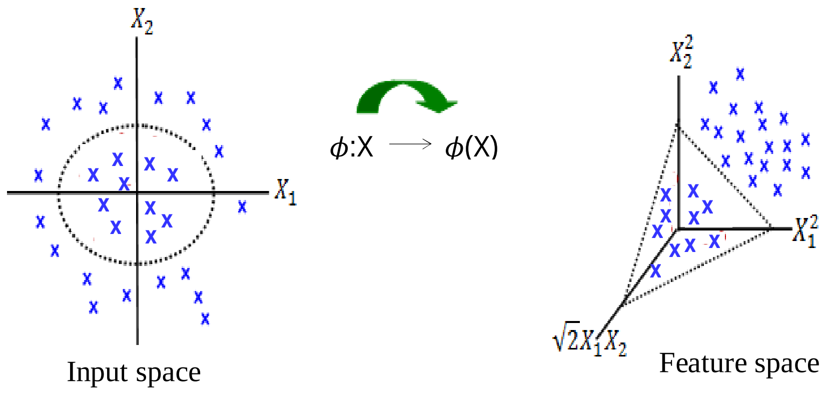


Figure 1.4: Mapping into Higher Dimensional Feature Space

As a generalization of linear SVR to nonlinear SVR, the basic idea is that input vector $\mathbf{x} \in R^n$ in the input space will be mapped into a higher dimensional Hilbert space called the feature space through a nonlinear mapping function $\phi(\mathbf{x})$ (B. Scholkopf et al, 1999; Aizerman et al., 1964; Boser et al., 1992). A linear regression function can be constructed in this feature space but it stays nonlinear in the input space. This is possible only with virtue of the Mercer's Theorem.

Most of the mapping functions $\phi(\mathbf{x})$ are unknown, but the dot product of the mapped vectors can be expressed as a function of the input vectors as

$$\phi(\mathbf{x}_1)^t \phi(\mathbf{x}_2) = k(\mathbf{x}_1, \mathbf{x}_2) \tag{1.15}$$

The feature spaces are called Reproducing Kernel Hilbert Spaces (RKHS), and $k(\cdot,\cdot)$ is a Mercer kernel. Fortunately, an explicit representation of the vectors in the feature space is not required as the SVM formulation only contains dot product of the mapped vectors.

The Mercer theorem gives the condition that a kernel function $k(\mathbf{x}_1, \mathbf{x}_2)$ must satisfy in order to be the dot product of a Hilbert space, i.e. there is a function ϕ in R^n such that $k(\mathbf{x}_1, \mathbf{x}_2) = \phi(\mathbf{x}_1)^t \phi(\mathbf{x}_2)$ if and only if for any function $g(\mathbf{x})$ for which

$$\int g(\mathbf{x})^2 d\mathbf{x} < \infty \qquad (1.16)$$

the inequality

$$\int k(\mathbf{x}, \mathbf{z}) g(\mathbf{x}) g(\mathbf{z}) d\mathbf{x} d\mathbf{z} \geq 0 \qquad (1.17)$$

holds.

Kernel functions must be symmetric and its Kernel matrix K is defined as

$$\mathrm{K} = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_m) \\ k(x_2, x_1) & k(x_2, x_1) & \vdots & k(x_2, x_m) \\ \vdots & \vdots & \vdots & \vdots \\ k(x_m, x_1) & k(x_m, x_1) & \cdots & k(x_m, x_m) \end{bmatrix}$$

The kernel matrix K is positive semi- definite (i.e. all its eigenvalues are non-negative i.e. $\lambda_i \geq 0, i = 1, \cdots, m$ and λ_i is an eigenvalue).

Kernel Function	**Kernel Definition** $k(\mathbf{x}_i, \mathbf{x}_j)$
Linear kernel	$\mathbf{x}_i^t \mathbf{x}_j$
Complete polynomial of degree d	$(1 + \mathbf{x}_i^t \mathbf{x}_j)^d, \quad d > 1$
Gaussian RBF	$\exp\left(-\frac{\lVert \mathbf{x}_i - \mathbf{x}_j \rVert^2}{2\sigma^2} \right), \quad \sigma > 0$
Sigmoidal	$\tanh(\gamma \mathbf{x}_i^t \mathbf{x}_j + \mu)^d, \gamma > 0, \mu > 0$

Table 1.2: Examples of Kernel functions used in SVM

The nonlinear mapping for polynomial kernels can be found in an explicit way and the corresponding Hilbert space has finite dimension. The nonlinear mapping for Gaussian kernel is not explicit and the dimension of Hilbert space is infinite.

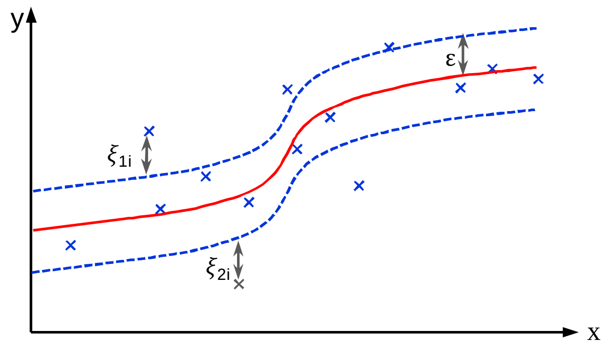


Figure 1.5: The geometrical interpretation of kernel SVR

For the nonlinear case, the kernel support vector regressor $(f : R^n \to R)$ is defined as

$$f(\mathbf{x}) = \mathbf{w}^t \phi(\mathbf{x}) + b \tag{1.18}$$

and will be obtained by the solving the following quadratic programming problem (Cristianini & Shawe-Taylor, 2000)

$$\min_{(\mathbf{w},b,\xi_1,\xi_2)\in R^{n+1+m+m}} \frac{1}{2}\mathbf{w}^t\mathbf{w} + C\sum_{i=1}^{m}(\xi_{1i}+\xi_{2i})$$

subject to

$$\begin{aligned} & y_i - (\mathbf{w}^t\phi(\mathbf{x}_i)+b) \le \varepsilon + \xi_{1i}, \\ & (\mathbf{w}^t\phi(\mathbf{x}_i)+b) - y_i \le \varepsilon + \xi_{2i}, \\ & \xi_{1i}, \xi_{2i} \ge 0, \quad \forall i = 1,2,\dots,m \end{aligned} \tag{1.19}$$

where ξ_{1i}, ξ_{2i} are slack variables, $\mathbf{y} = (y_1, \cdots, y_m)^t$ is output vector and $C > 0, \varepsilon > 0$ are input parameters.

Proceeding as in linear SVR and using kernel trick (1.15), the dual of (1.19) can be obtained as

$$\min_{\mathbf{u}_1,\mathbf{u}_2\in R^m} \frac{1}{2}\sum_{i,j=1}^{m}(u_{1i}-u_{2i})k(\mathbf{x}_i,\mathbf{x}_j)(u_{1j}-u_{2j}) - \sum_{i=1}^{m} y_i(u_{1i}-u_{2i}) + \varepsilon\sum_{i=1}^{m}(u_{1i}-u_{2i}) \tag{1.20}$$

subject to

$$\mathbf{e}^t(\mathbf{u}_1-\mathbf{u}_2)=0 \text{ and } \mathbf{0}\le \mathbf{u}_1,\mathbf{u}_2 \le C\mathbf{e}$$

Finally, for any sample $\mathbf{x}\in R^n$ the nonlinear regressor (1.18) becomes

$$f(\mathbf{x}) = \sum_{i=1}^{m}(u_{1i}-u_{2i})k(\mathbf{x},\mathbf{x}_i) + b \tag{1.21}$$

Chapter 2

Twin Support Vector Machine in Regression

2.1 Introduction

Support Vector Machines (SVMs) have been extensively studied and applied to a number of classification and regression problems which have shown remarkable success compared to other machine learning methods such as ANNs. SVMs show distinct advantages such as better generalization, the ability to find a global optimum, and the increased speed of learning. However, training an SVM involves solving a constrained quadratic programming problem (CQPP). Its training computational complexity is $\mathrm{O}(m^3)$, where m is the total size of training set. This means much increased computational time for large dataset.

In order to speed up the training process of SVM, many efforts have been made such as chunking and decomposition methods (Boser et al., 1992; Joachims, 1999; Kaufman, 1999; Osuna et al., 1997), exact SVM training algorithm SMO (Platt, 1999), approximate SVM training algorithms (Tsang et al., 2005; Achlioptas et al., 2002; Fine et al., 2001), LS-SVM (Suykens & Vandewalle, 1999; Suykens, Lukas, Van Dooren, et al., 1999), etc. The above algorithms solve the dual of CQPP iteratively and at each step of iteration only a subset of the dual variables are optimized. Recently Twin Support Vector Machine (TWSVM) has been proposed (Jaydeva et al., 2007) by extending the work of GEPSVM (Mangasarian & Wild, 2006) in which two nonparallel planes are constructed such that each plane is closer to one of the two classes and is as far as possible from the other. The performance of TWSVM is better than GEPSVM and is approximately four times faster than SVM.

As for SVR, there exist some corresponding approximation algorithms as in classification, such as Smooth SVR (Lee et al, 2005), SMO (Shevade et al, 2000), etc. A fast training algorithm known as Twin Support Vector Regression (TSVR) (Peng, 2010) is described in the next section 2.2. Most recently Twin Support Vector Machine Based

Regression (TWSVR) proposed in (Khemchandani, Goyal and Chandra, 2015) overcomes the restrictions associated with TSVR will be introduced in the next Chapter.

2.2 Twin Support Vector Regression (TSVR)

In the spirit of TWSVM, an efficient twin support vector regression, termed as TSVR, is proposed in (Peng, 2010) for regression problem to improve the computational training speed. Assume that we are given a training dataset (1.1). The TSVR generates a pair of nonparallel hyperplanes such that one of them determines the ε-insensitive down bound $f_1(\mathbf{x}) = \mathbf{w}_1^t\mathbf{x} + b_1$ and another one the upper bound function $f_2(\mathbf{x}) = \mathbf{w}_2^t\mathbf{x} + b_2$ of the end regressor. Similar to the idea of maximum margin, these hyperplanes are constructed to be as far as possible from each other.

The final regressor is obtained by taking the mean of these functions as follows:

$$f(\mathbf{x}) = \frac{1}{2}(f_1(\mathbf{x}) + f_2(\mathbf{x})) = \frac{1}{2}(\mathbf{w}_1 + \mathbf{w}_2)^t\mathbf{x} + \frac{1}{2}(b_1 + b_2)$$

$$\text{i.e.} \quad f(\mathbf{x}) = \frac{1}{2}([\mathbf{w}_1 \quad b_1] + [\mathbf{w}_2 \quad b_2])\begin{bmatrix} x \\ 1 \end{bmatrix} \qquad (2.1)$$

2.2.1 Linear Twin Support Vector Regression

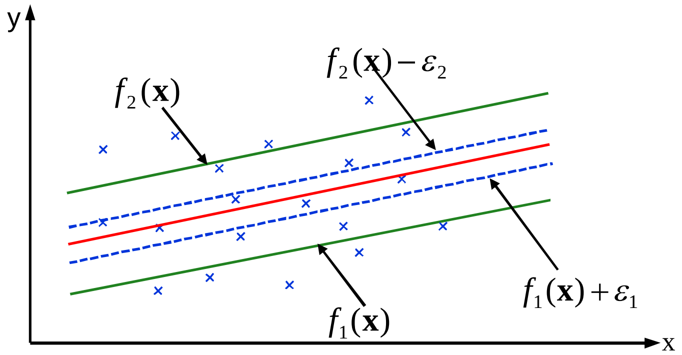

y
$f_2(\mathbf{x})$
$f_2(\mathbf{x}) - \varepsilon_2$
$f_1(\mathbf{x}) + \varepsilon_1$
$f_1(\mathbf{x})$
x

Figure 2.1: Geometrical interpretation of linear TSVR

It is well known that, TSVR constructs two nonparallel hyperplanes in the input space (see Figure 2.1) defined as

$$f_1(\mathbf{x}) = \mathbf{w}_1^t\mathbf{x} + b_1 \quad \text{and} \quad f_2(\mathbf{x}) = \mathbf{w}_2^t\mathbf{x} + b_2 \qquad (2.2)$$

These hyperplanes are determined by solving the following pair of constrained quadratic programming problems (CQPPs):

$$\min_{(\mathbf{w}_1,b_1,\boldsymbol{\xi}_1)\in R^{(n+1+m)}} \frac{1}{2}(\mathbf{y}-\mathbf{e}\varepsilon_1-(A\mathbf{w}_1+\mathbf{e}b_1))^t(\mathbf{y}-\mathbf{e}\varepsilon_1-(A\mathbf{w}_1+\mathbf{e}b_1))+C_1\mathbf{e}^t\boldsymbol{\xi}_1$$

subject to $\mathbf{y}-(A\mathbf{w}_1+\mathbf{e}b_1)\ge \mathbf{e}\varepsilon_1-\boldsymbol{\xi}_1,\quad \boldsymbol{\xi}_1\ge\mathbf{0}$ (2.3)

and $$\min_{(\mathbf{w}_2,b_2,\boldsymbol{\xi}_2)\in R^{(n+1+m)}} \frac{1}{2}(\mathbf{y}+\mathbf{e}\varepsilon_2-(A\mathbf{w}_2+\mathbf{e}b_2))^t(\mathbf{y}+\mathbf{e}\varepsilon_2-(A\mathbf{w}_2+\mathbf{e}b_2))+C_2\mathbf{e}^t\boldsymbol{\xi}_2$$

subject to $(A\mathbf{w}_2+\mathbf{e}b_2)-\mathbf{y}\ge \mathbf{e}\varepsilon_2-\boldsymbol{\xi}_2,\quad \boldsymbol{\xi}_2\ge\mathbf{0}$ (2.4)

where $C_1,C_2>0,\varepsilon_1,\varepsilon_2\ge 0$ are input parameters; $\boldsymbol{\xi}_1=(\xi_{11},\cdots,\xi_{1m})^t$, $\boldsymbol{\xi}_2=(\xi_{21},\cdots,\xi_{2m})^t$ are vectors of slack variables and the training samples are organized in matrix A whose ith row A_i becomes $\mathbf{x}_i^t$.

Introducing Lagrange multipliers $\boldsymbol{\alpha}_1=(\alpha_{11},\cdots,\alpha_{1m})^t,\boldsymbol{\beta}_1=(\beta_{11},\cdots,\beta_{1m})^t$ and $\boldsymbol{\alpha}_2=(\alpha_{21},\cdots,\alpha_{2m})^t,\boldsymbol{\beta}_2=(\beta_{21},\cdots,\beta_{2m})^t$ for CQPPs (2.3) and (2.4), their Lagrangian functions can be written as:

$$L_1(\mathbf{w}_1,b_1,\boldsymbol{\xi}_1,\boldsymbol{\alpha}_1,\boldsymbol{\beta}_1)=\frac{1}{2}(\mathbf{y}-\mathbf{e}\varepsilon_1-(A\mathbf{w}_1+eb_1))^t\times(\mathbf{y}-\mathbf{e}\varepsilon_1-(A\mathbf{w}_1+\mathbf{e}b_1))+C_1\mathbf{e}^t\boldsymbol{\xi}_1$$
$$-\boldsymbol{\alpha}_1^t(\mathbf{y}-(A\mathbf{w}_1+\mathbf{e}b_1)-\mathbf{e}\varepsilon_1+\boldsymbol{\xi}_1))-\boldsymbol{\beta}_1^t\boldsymbol{\xi}_1 \quad (2.5)$$

$$L_2(\mathbf{w}_2,b_2,\boldsymbol{\xi}_2,\boldsymbol{\alpha}_2,\boldsymbol{\beta}_2)=\frac{1}{2}(\mathbf{y}+\mathbf{e}\varepsilon_2-(A\mathbf{w}_2+\mathbf{e}b_2))^t\times(\mathbf{y}+\mathbf{e}\varepsilon_2-(A\mathbf{w}_2+\mathbf{e}b_2))+C_2\mathbf{e}^t\boldsymbol{\xi}_2$$
$$-\boldsymbol{\alpha}_2^t((\mathbf{A}\mathbf{w}_2+\mathbf{e}b_2)-\mathbf{y}-\mathbf{e}\varepsilon_2+\boldsymbol{\xi}_2))-\boldsymbol{\beta}_2^t\boldsymbol{\xi}_2 \quad (2.6)$$

Using the KKT conditions for Lagrangian function (2.5), we get:

$$-\mathbf{A}^t(\mathbf{y}-\mathbf{e}\varepsilon_1-\mathbf{A}\mathbf{w}_1-\mathbf{e}b_1)+\mathbf{A}^t\boldsymbol{\alpha}=\mathbf{0} \quad (2.7)$$

$$-\mathbf{e}^t(\mathbf{y}-\mathbf{e}\varepsilon_1-\mathbf{A}\mathbf{w}_1-\mathbf{e}b_1)+\mathbf{e}^t\boldsymbol{\alpha}_1=0 \quad (2.8)$$

$$C_1\mathbf{e}-\boldsymbol{\alpha}_1-\boldsymbol{\beta}_1=\mathbf{0} \quad (2.9)$$

$$\mathbf{y}-\mathbf{A}\mathbf{w}_1-\mathbf{e}b_1\ge\mathbf{e}\varepsilon_1-\boldsymbol{\xi}_1,\quad \boldsymbol{\xi}_1\ge\mathbf{0} \quad (2.10)$$

$$\boldsymbol{\alpha}_1^t((\mathbf{y}-\mathbf{A}\mathbf{w}_1-\mathbf{e}b_1)-\mathbf{e}\varepsilon_1+\boldsymbol{\xi}_1)=0,\quad \boldsymbol{\alpha}_1\ge\mathbf{0} \quad (2.11)$$

$$\boldsymbol{\beta}_1^t\boldsymbol{\xi}_1=0,\quad \boldsymbol{\beta}_1\ge\mathbf{0} \quad (2.12)$$

Since $\boldsymbol{\beta}_1\ge\mathbf{0}$, from (2.9) we have

$$\mathbf{0}\le\boldsymbol{\alpha}_1\le C_1\mathbf{e} \quad (2.13)$$

Similarly for Lagrangian function (2.6), we get

$$-\mathbf{A}^t(\mathbf{y}+\mathbf{e}\varepsilon_2-\mathbf{A}\mathbf{w}_2-\mathbf{e}b_2)-\mathbf{A}^t\boldsymbol{\alpha}_2=\mathbf{0} \tag{2.14}$$

$$-\mathbf{e}^t(\mathbf{y}+\mathbf{e}\varepsilon_2-\mathbf{A}\mathbf{w}_2-\mathbf{e}b_2)-\mathbf{e}^t\boldsymbol{\alpha}_2=0 \tag{2.15}$$

$$C_2\mathbf{e}-\boldsymbol{\alpha}_2-\boldsymbol{\beta}_2=\mathbf{0} \tag{2.16}$$

$$(\mathbf{A}\mathbf{w}_2+\mathbf{e}b_2)-\mathbf{y}\geq\mathbf{e}\varepsilon_2-\boldsymbol{\xi}_2,\quad \boldsymbol{\xi}_2\geq\mathbf{0} \tag{2.17}$$

$$\boldsymbol{\alpha}_2^t((\mathbf{A}\mathbf{w}_2+\mathbf{e}b_2)-\mathbf{y}-\mathbf{e}\varepsilon_2+\boldsymbol{\xi}_2)=0,\quad \boldsymbol{\alpha}_2\geq\mathbf{0} \tag{2.18}$$

$$\boldsymbol{\beta}_2^t\boldsymbol{\xi}_2=0,\quad \boldsymbol{\beta}_2\geq\mathbf{0} \tag{2.19}$$

Since $\boldsymbol{\beta}_2\geq 0$, from (2.16) we

$$\mathbf{0}\leq\boldsymbol{\alpha}_2\leq C_2\mathbf{e},$$

(2.20)

Now, combining (2.7) with (2.8) and (2.14) with (2.15), we get

$$-\begin{bmatrix}\mathbf{A}^t\\ \mathbf{e}^t\end{bmatrix}\left((\mathbf{y}-\mathbf{e}\varepsilon_1)-[\mathbf{A}\quad\mathbf{e}]\begin{bmatrix}\mathbf{w}_1\\ b_1\end{bmatrix}\right)+\begin{bmatrix}\mathbf{A}^t\\ e^t\end{bmatrix}\boldsymbol{\alpha}_1=0 \tag{2.21}$$

$$-\begin{bmatrix}\mathbf{A}^t\\ \mathbf{e}^t\end{bmatrix}\left((\mathbf{y}+\mathbf{e}\varepsilon_1)-[\mathbf{A}\quad\mathbf{e}]\begin{bmatrix}\mathbf{w}_2\\ b_2\end{bmatrix}\right)-\begin{bmatrix}\mathbf{A}^t\\ e^t\end{bmatrix}\boldsymbol{\alpha}_2=0 \tag{2.22}$$

For the sake of convenience in expression, Equations (2.21) and (2.22) can be written in the following simpler forms, i.e.

$$-G^t\mathbf{f}_1+G^tG\mathbf{u}_1+G^t\boldsymbol{\alpha}_1=0 \qquad \text{i.e. } \mathbf{u}_1=(G^tG)^{-1}G^t(\mathbf{f}_1-\boldsymbol{\alpha}_1). \tag{2.23}$$

$$-G^t\mathbf{f}_2+G^tG\mathbf{u}_2-G^t\boldsymbol{\alpha}_2=0 \qquad \text{i.e. } \mathbf{u}_2=(G^tG)^{-1}G^t(\mathbf{f}_2+\boldsymbol{\alpha}_2). \tag{2.24}$$

where $G=[\mathbf{A}\quad\mathbf{e}], \mathbf{f}_1=\mathbf{y}-\mathbf{e}\varepsilon_1, \mathbf{u}_1=[\mathbf{w}_1^t\quad b_1]^t, \mathbf{f}_2=\mathbf{y}+\mathbf{e}\varepsilon_2,$ and $\mathbf{u}_2=[\mathbf{w}_2^t\quad b_2]^t.$ Note that G^tG is positive semidefinite but in order to overcome the situations in which its inverse may not exist, a regularization term σI is introduced so that $(G^tG+\sigma I)$ becomes positive definite where σ is a very small positive number $(\sigma=1e-7).$ Thus we have

$$\mathbf{u}_1=(G^tG+\sigma I)^{-1}G^t(\mathbf{f}_1-\boldsymbol{\alpha}_1), \tag{2.25}$$

$$\mathbf{u}_2=(G^tG+\sigma I)^{-1}G^t(\mathbf{f}_2+\boldsymbol{\alpha}_2). \tag{2.26}$$

Substituting (2.23) in the primal Lagrangian function (2.5) and using (2.10) to

(2.13), the dual CQPP of (2.3) can be obtained as

$$\min_{\boldsymbol{\alpha}_1 \in R^m} \quad \frac{1}{2}\boldsymbol{\alpha}_1^t G(G^t G)^{-1} G^t \boldsymbol{\alpha}_1 - \mathbf{f}_1^t G(G^t G)^{-1} G^t \boldsymbol{\alpha}_1 + \mathbf{f}_1^t \boldsymbol{\alpha}_1$$

subject to $\quad \mathbf{0} \le \boldsymbol{\alpha}_1 \le C_1 \mathbf{e}.$ (2.27)

Similarly, substituting (2.24) in the primal Lagrangian function (2.6) and using (2.17) to (2.20), the dual CQPP of (2.4) can be obtained as

$$\min_{\boldsymbol{\alpha}_2 \in R^m} \quad \frac{1}{2}\boldsymbol{\alpha}_2^t G(G^t G)^{-1} G^t \boldsymbol{\alpha}_2 + \mathbf{f}_2^t G(G^t G)^{-1} G^t \boldsymbol{\alpha}_2 - \mathbf{f}_2^t \boldsymbol{\alpha}_2$$

subject to $\quad \mathbf{0} \le \boldsymbol{\alpha}_2 \le C_2 \mathbf{e}.$ (2.28)

Once the vectors $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$ are known, by solving (2.27) and (2.28) the output for any data point $\mathbf{x} \in R^n$ is predicted by

$$f(\mathbf{x}) = \frac{1}{2}(f_1(\mathbf{x}) + f_2(\mathbf{x})) = \frac{1}{2}([\mathbf{w}_1 \quad b_1] + [\mathbf{w}_2 \quad b_2])^t [\mathbf{x} \quad 1]^t = \frac{1}{2}[\mathbf{x}^t \quad 1](\mathbf{u}_1 + \mathbf{u}_2)\,.$$

2.2.2 Kernel Twin Support Vector Regression

For the nonlinear case, let the input vectors $\mathbf{x} \in R^n$ be mapped into a high dimensional feature space through a nonlinear mapping function $\phi(\mathbf{x})$. Assume that the dot product of any two vectors $\phi(\mathbf{x}_1), \phi(\mathbf{x}_2)$ is given by

$$k(\mathbf{x}_1, \mathbf{x}_2) = \phi(\mathbf{x}_1)^t \, \phi(\mathbf{x}_2)$$

where $k(.,.)$ is any suitable kernel function (see Table 1.2). Define the kernel matrix $K = K(A, A^t)$ of size $m \times m$ whose $(i,j)^{th}$ entry is $k(\mathbf{x}_i, \mathbf{x}_j)$ and also let $K(\mathbf{x}^t, A^t) = (k(\mathbf{x}, \mathbf{x}_1) \quad \cdots \quad k(\mathbf{x}, \mathbf{x}_m))$ be a row vector.

In this case, the ε-insensitive down bound and up bound functions are defined by (Peng, 2010)

$$f_1(\mathbf{x}) = K(\mathbf{x}^t, A^t)\mathbf{w}_1 + b_1, \quad f_2(\mathbf{x}) = K(\mathbf{x}^t, A^t)\mathbf{w}_2 + b_2 \qquad (2.29)$$

where $\mathbf{w}_1, \mathbf{w}_2 \in R^m$. An intuitive geometric interpretation of nonlinear TSVR is shown in Figure 2.2.

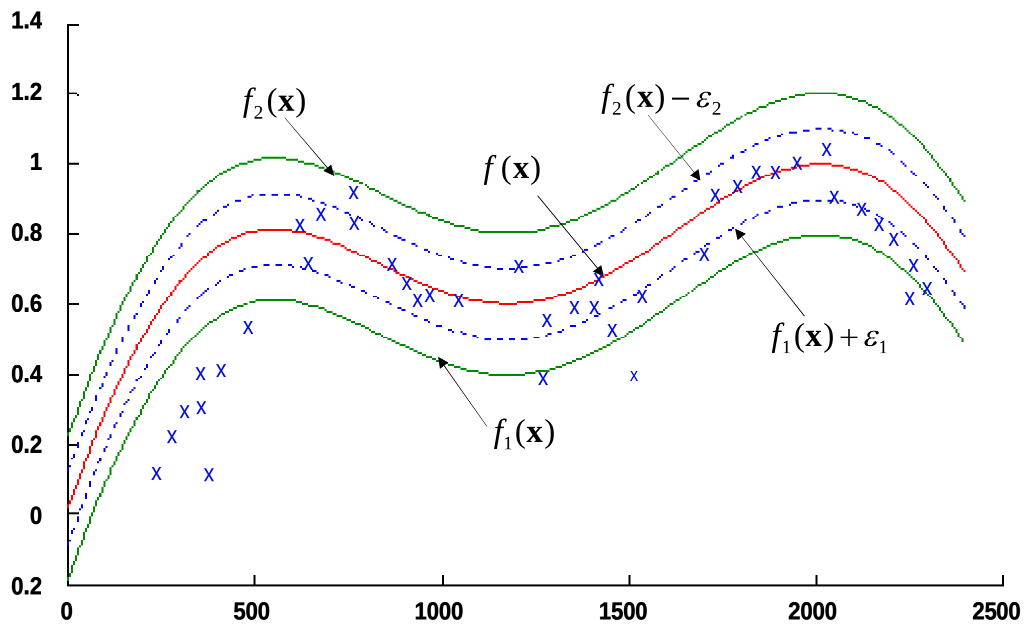


Figure 2.2: Geometrical interpretation of kernel TSVR

These hyperplanes are determined by solving the following pairs of CQPPs

$$\min_{(\mathbf{w}_1,b_1,\xi_1)\in R^{(n+1+m)}} \frac{1}{2}(\mathbf{y}-\mathbf{e}\varepsilon_1-(K(A,A^t)\mathbf{w}_1+\mathbf{e}b_1))^t\times(\mathbf{y}-\mathbf{e}\varepsilon_1-(K(A,A^t)\mathbf{w}_1+\mathbf{e}b_1))$$
$$+C_1\mathbf{e}^t\boldsymbol{\xi}_1$$

subject to

$$\mathbf{y}-(K(A,A^t)\mathbf{w}_1+\mathbf{e}b_1)\geq\mathbf{e}\varepsilon_1-\boldsymbol{\xi}_1,\quad \boldsymbol{\xi}_1\geq\mathbf{0} \tag{2.30}$$

and

$$\min_{(\mathbf{w}_2,b_2,\xi_2)\in R^{(n+1+m)}} \frac{1}{2}(\mathbf{y}+\mathbf{e}\varepsilon_2-(K(A,A^t)\mathbf{w}_2+\mathbf{e}b_2))^t\times(\mathbf{y}+\mathbf{e}\varepsilon_2-(K(A,A^t)\mathbf{w}_2+\mathbf{e}b_2))$$
$$+C_2\mathbf{e}^t\boldsymbol{\xi}_2$$

subject to

$$(K(A,A^t)\mathbf{w}_2+\mathbf{e}b_2)-\mathbf{y}\geq\mathbf{e}\varepsilon_2-\boldsymbol{\xi}_2,\quad \boldsymbol{\xi}_2\geq\mathbf{0} \tag{2.31}$$

Introducing Lagrange multipliers $\boldsymbol{\alpha}_1=(\alpha_{11},\cdots,\alpha_{1m})^t,\boldsymbol{\beta}_1=(\beta_{11},\cdots,\beta_{1m})^t$, $\boldsymbol{\alpha}_2=(\alpha_{21},\cdots,\alpha_{2m})^t$ and $\boldsymbol{\beta}_2=(\beta_{21},\cdots,\beta_{2m})^t$ the Lagrangian functions corresponding to (2.30) and (2.31) become

$$L_1(\mathbf{w}_1, b_1, \boldsymbol{\xi}_1, \boldsymbol{\alpha}_1, \boldsymbol{\beta}_1) = \frac{1}{2}(\mathbf{y} - \mathbf{e}\varepsilon_1 - (K(A, A^t)\mathbf{w}_1 + \mathbf{e}b_1))^t \times (\mathbf{y} - \mathbf{e}\varepsilon_1 - (K(A, A^t)\mathbf{w}_1 + \mathbf{e}b_1)$$
$$+ C_1\mathbf{e}^t\boldsymbol{\xi}_1 - \boldsymbol{\alpha}_1^t(\mathbf{y} - (K(A, A^t)\mathbf{w}_1 + \mathbf{e}b_1) - \mathbf{e}\varepsilon_1 + \boldsymbol{\xi}_1) - \boldsymbol{\beta}_1^t\boldsymbol{\xi}_1 \qquad (2.32)$$

and

$$L_2(\mathbf{w}_2, b_2, \boldsymbol{\xi}_2, \boldsymbol{\alpha}_2, \boldsymbol{\beta}_2) = \frac{1}{2}(\mathbf{y} + \mathbf{e}\varepsilon_2 - (K(A, A^t)\mathbf{w}_2 + \mathbf{e}b_2))^t \times (\mathbf{y} + \mathbf{e}\varepsilon_2 - (K(A, A^t)\mathbf{w}_2 + \mathbf{e}b_2))$$
$$+ C_2\mathbf{e}^t\boldsymbol{\xi}_2 - \boldsymbol{\alpha}_2^t((K(A, A^t)\mathbf{w}_2 + \mathbf{e}b_2) - \mathbf{y} - \mathbf{e}\varepsilon_2 + \boldsymbol{\xi}_2) - \boldsymbol{\beta}_2^t\boldsymbol{\xi}_2 \qquad (2.33)$$

respectively.

Proceeding as we have done for the linear case and taking $[\mathbf{w}_1^t \quad b_1]^t = \mathbf{u}_1$ and $[\mathbf{w}_2^t \quad b_2]^t = \mathbf{u}_2$, the duals of (2.30) and (2.31) can be obtained in the following forms:

$$\min_{\boldsymbol{\alpha}_1 \in R^m} \quad \frac{1}{2}\boldsymbol{\alpha}_1^t G(G^tG)^{-1}G^t\boldsymbol{\alpha}_1 - \mathbf{f}_1^t G(G^tG)^{-1}G^t\boldsymbol{\alpha}_1 + \mathbf{f}_1^t\boldsymbol{\alpha}_1$$

subject to $\quad \mathbf{0} \le \boldsymbol{\alpha}_1 \le C_1\mathbf{e},$ (2.34)

and

$$\min_{\boldsymbol{\alpha}_2 \in R^m} \quad \frac{1}{2}\boldsymbol{\alpha}_2^t G(G^tG)^{-1}G^t\boldsymbol{\alpha}_2 + \mathbf{f}_2^t G(G^tG)^{-1}G^t\boldsymbol{\alpha}_2 - \mathbf{f}_2^t\boldsymbol{\alpha}_2$$

subject to $\quad \mathbf{0} \le \boldsymbol{\alpha}_2 \le C_2\mathbf{e},$ (2.35)

respectively, where $G = [K(A, A^t) \quad \mathbf{e}]$.

Further, we have

$$\mathbf{u}_1 = (G^tG + \sigma I)^{-1}G^t(\mathbf{f}_1 - \boldsymbol{\alpha}_1), \qquad (2.36)$$

$$\mathbf{u}_2 = (G^tG + \sigma I)^{-1}G^t(\mathbf{f}_2 + \boldsymbol{\alpha}_2). \qquad (2.37)$$

For an unknown sample $\mathbf{x} \in R^n$, its prediction becomes

$$f(\mathbf{x}) = \frac{1}{2}(f_1(\mathbf{x}) + f_2(\mathbf{x})) = \frac{1}{2}[K(\mathbf{x}^t, A^t) \quad 1](\mathbf{u}_1 + \mathbf{u}_2)$$

Chapter 3

Twin Support Vector Machine Based Regression

3.1 Introduction

In this chapter, we introduce TWSVM Based Regression (TWSVR) proposed by Khemchandani, Goyal and Chandra (2015). This study was inspired from the work done by Bi and Bennett (2003) where they have given geometrical interpretation on how a SVR problem can be regarded as a classification problem. The end regressor $f(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + b$ is the average of two nonparallel hyperplanes i.e. ε_1-insensitive down bound and ε_2-insensitive up bound regressors determined by solving a pair of CQPPs similar to TSVR (Peng, 2010). They claimed that though Peng's approach (2010) to TSVR was motivated from TWSVM but its formulation is not on the lines of TWSVM and the parameters ε_1 and ε_2 affect the linear shift of the end regressor $f(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + b.$ More precisely, only b depends on the ε_1 and ε_2 but $\mathbf{w}$ is independent of the values of ε_1 and ε_2.

TWSVR formulation has been mathematically derived from the TWSVM (Khemchandani et al., 2015) as the standard SVR is related to SVM (Bi and Bennett, 2003). Unlike TSVR, both the parameters ε_1 and ε_2 contribute in the orientation of the end regressor $f(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + b$ i.e. $\mathbf{w}$ and b are functions of both ε_1 and ε_2. For the standard SVR, epsilon also contributes in the orientation of regressor. This means value of epsilon not only contributes to linear shift of regressor from origin but also determines the end regressor.

Like TSVR, TWSVR also provides improved results than the standard SVR and is approximately four times faster than standard SVR (Peng, 2010). Their formulations differ in the ε term only. TWSVR also achieves comparable results to TSVR because ε is chosen to be a small quantity. The formulation of TWSVR is not only better than TSVR but also is the correct choice for future work on TWSVR (Khemchandani et al., 2015).

3.2 Linear TWSVR

In this section we briefly introduce how a regressor problem (SVR, TWSVR) can be regarded as classification problem (SVM, TWSVM) (Bi & Bennett, 2003; Khemchandani et al., 2015) . Assume that we are given a training dataset (1.1) i.e.

$$S = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_m, y_m)\}, \quad \mathbf{x}_i = (x_{i1}, \cdots, x_{in})^t \in X \subseteq R^n, y_i \in Y \subseteq R.$$

Let D^+ and D^- be formed by shifting up and down output variables of training points by $\varepsilon' > 0$ i.e. $D^+ = \{(\mathbf{x}_i, y_i + \varepsilon'), i = 1, \cdots, m\}$ and $D^- = \{(\mathbf{x}_i, y_i - \varepsilon'), i = 1, \cdots, m\}$ .These can also be written as

$$D^+ = \{(A_i, y_i + \varepsilon'), i = 1, \cdots, m\} \tag{3.1}$$

$$D^- = \{(A_i, y_i - \varepsilon'), i = 1, \cdots, m\} . \tag{3.2}$$

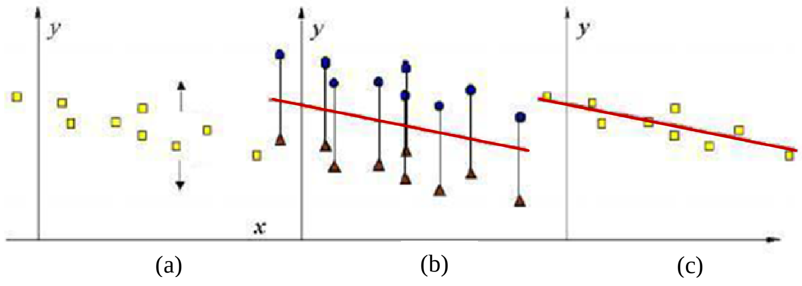


Figure 3.1: SVM Regression; (a) original data (b) shifted data and separating hyperplane (c) regression plane (Bi & Bennett, 2003).

In the augmented space $n+1$, we assign label +1 and -1 to training points of D^+ and D^- respectively. We find a hyperplane separating these two classes of samples that can be considered as the regressor function in the n dimensional space (input space).Thus the problem of finding a SVR hyperplane in n dimensional space is equivalent to finding a SVM hyperplane in $n+1$ dimensional space.

In case of TWSVR, TWSVM method instead of SVM is applied on the two sets D^+ and D^- that determines two hyperplanes, one close to D^+ and other to D^-. These hyperplanes become up bound and down bound function of the end regressor in the input space, where the end regressor is their average.

3.2.1 Hard classifier Linear ε -Insensitive TWSVR

In this section, we introduce the derivation of the TWSVR from the formulations of TWSVM having no error term in their objective functions. We apply TWSVM on the two sets $D^+,\ D^-$ (see section 3.2) and obtain two nonparallel hyperplanes as the solutions of the QPPs

$$\min_{\mathbf{w}_1, b_1, \eta_1} \ \frac{1}{2}\left\| A\mathbf{w}_1 + \eta_1(\mathbf{y} + \mathbf{e}\varepsilon') + \mathbf{e}b_1 \right\|^2$$

subject to $\quad (A\mathbf{w}_1 + \eta_1(\mathbf{y} - \mathbf{e}\varepsilon') + \mathbf{e}b_1) + \mathbf{e} \le \mathbf{0} \qquad (3.3)$

$$\min_{\mathbf{w}_2, b_2, \eta_2} \ \frac{1}{2}\left\| A\mathbf{w}_2 + \eta_2(\mathbf{y} - \mathbf{e}\varepsilon') + \mathbf{e}b_2 \right\|^2$$

subject to $\quad (A\mathbf{w}_2 + \eta_2(\mathbf{y} + \mathbf{e}\varepsilon') + \mathbf{e}b_2) - \mathbf{e} \ge \mathbf{0} \qquad (3.4)$

Assume that the solution of (3.3) and (3.4) determines two hyperplanes $\mathbf{w}_1'^t\mathbf{x} + \eta_1'^t y + b_1' = 0$ and $\mathbf{w}_2'^t\mathbf{x} + \eta_2'^t y + b_2' = 0$. Now we fix η_i' for $i = 1,2$ and apply the following transformations (a) to (b) on above formulations as follows:

a) Assuming $\eta_i' > 0$ and replacing $\mathbf{w}_i = -\mathbf{w}_i / \eta_i',\ b_i = b_i / \eta_i'$ (3.3) and (3.4) become

$$\min_{\mathbf{w}_1, b_1, \eta_1} \ \frac{1}{2}\left(\mathbf{y} + \mathbf{e}\varepsilon' - (A\mathbf{w}_1 + \mathbf{e}b_1)\right)^t \left(\mathbf{y} + \mathbf{e}\varepsilon' - (A\mathbf{w}_1 + \mathbf{e}b_1)\right)$$

subject to $\quad (A\mathbf{w}_1 + \mathbf{e}b_1) \ge \mathbf{y} - \mathbf{e}\varepsilon' + \frac{1}{\eta_1'}\mathbf{e}$

$$\min_{\mathbf{w}_2, b_2, \eta_2} \ \frac{1}{2}\left(\mathbf{y} - \mathbf{e}\varepsilon' - (A\mathbf{w}_2 + \mathbf{e}b_2)\right)^t \left(\mathbf{y} - \mathbf{e}\varepsilon' - (A\mathbf{w}_2 + \mathbf{e}b_2)\right)$$

subject to $\quad (A\mathbf{w}_2 + \mathbf{e}b_2) \le \mathbf{y} + \mathbf{e}\varepsilon' - \frac{1}{\eta_2'}\mathbf{e}$

b) Subtract and add by ε' on the both sides of first and second CQPP respectively:

$$\min_{\mathbf{w}_1, b_1, \eta_1} \ \frac{1}{2}\left(\mathbf{y} + \mathbf{e}\varepsilon' - (A\mathbf{w}_1 + \mathbf{e}b_1)\right)^t \left(\mathbf{y} + \mathbf{e}\varepsilon' - (A\mathbf{w}_1 + \mathbf{e}b_1)\right)$$

subject to $\quad (A\mathbf{w}_1 + \mathbf{e}b_1) - \mathbf{e}\varepsilon' \ge \mathbf{y} - \left(2\mathbf{e}\varepsilon' - (1/\eta_1')\mathbf{e}\right)$

$$\min_{\mathbf{w}_2, b_2, \eta_2} \quad \frac{1}{2}\left(\mathbf{y} - \varepsilon' \mathbf{e} - (A\mathbf{w}_2 + \mathbf{e}b_2)\right)^t \left(\mathbf{y} - \varepsilon' \mathbf{e} - (A\mathbf{w}_2 + eb_2)\right)$$

subject to $(A\mathbf{w}_2 + \mathbf{e}b_2) + \mathbf{e}\varepsilon' \leq \mathbf{y} + \left(2\mathbf{e}\varepsilon' - (1/\eta_2')\mathbf{e}\right)$

c) Replace $b_1\mathbf{e} = b_1\mathbf{e} - \mathbf{e}\varepsilon'$ and $b_2\mathbf{e} = b_2\mathbf{e} + \mathbf{e}\varepsilon'$ on the first and second CQPP.

$$\min_{\mathbf{w}_1, b_1, \eta_1} \quad \frac{1}{2}\left(\mathbf{y} - (A\mathbf{w}_1 + \mathbf{e}b_1)\right)^t \left(\mathbf{y} - (A\mathbf{w}_1 + \mathbf{e}b_1)\right)$$

subject to $(A\mathbf{w}_1 + \mathbf{e}b_1) \geq \mathbf{y} - \left(2\mathbf{e}\varepsilon' - (1/\eta_1')\mathbf{e}\right)$

$$\min_{\mathbf{w}_2, b_2, \eta_2} \quad \frac{1}{2}\left(\mathbf{y} - (A\mathbf{w}_2 + \mathbf{e}b_2)\right)^t \left(\mathbf{y} - (A\mathbf{w}_2 + \mathbf{e}b_2)\right)$$

subject to $(A\mathbf{w}_2 + \mathbf{e}b_2) \leq \mathbf{y} + \left(2\mathbf{e}\varepsilon' - (1/\eta_2')\mathbf{e}\right)$

d) Apply transformation $\varepsilon_i = 2\varepsilon' - (1/\eta_i')$ such that $\varepsilon_i \geq 0$:

$$\min_{\mathbf{w}_1, b_1} \quad \frac{1}{2}\left(\mathbf{y} - (A\mathbf{w}_1 + \mathbf{e}b_1)\right)^t \left(\mathbf{y} - (A\mathbf{w}_1 + \mathbf{e}b_1)\right)$$

subject to $\mathbf{y} - (A\mathbf{w}_1 + \mathbf{e}b_1) \leq \mathbf{e}\varepsilon_1$ (3.5)

$$\min_{\mathbf{w}_2, b_2} \quad \frac{1}{2}\left(\mathbf{y} - (A\mathbf{w}_2 + \mathbf{e}b_2)\right)^t \left(\mathbf{y} - (A\mathbf{w}_2 + \mathbf{e}b_2)\right)$$

subject to $(A\mathbf{w}_2 + \mathbf{e}b_2) - \mathbf{y} \leq \mathbf{e}\varepsilon_2$ (3.6)

Considering these transformations, the solutions $(-\mathbf{w}_1'/\eta_1', -b_1'/\eta_1' - \varepsilon')$ and $(-\mathbf{w}_2'/\eta_2', -b_2'/\eta_2' + \varepsilon')$ of (3.5) and (3.6) determine the two regressor hyperplanes

$$f_1(\mathbf{x}) = \mathbf{w}_1^*\mathbf{x} + b_1^* \text{ and } f_2(\mathbf{x}) = \mathbf{w}_2^*\mathbf{x} + b_2^*,$$

where $\mathbf{w}_1^* = -\mathbf{w}_1'/\eta_1'$, $\mathbf{w}_2^* = -\mathbf{w}_2'/\eta_2'$, $b_1^* = -b_1'/\eta_1' - \varepsilon'$ and $b_2^* = -b_2'/\eta_2' + \varepsilon'$.

The final regressor $f(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + b$ is obtained as the mean of $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$, where $\mathbf{w} = (-\mathbf{w}_1'/\eta_1' - \mathbf{w}_2'/\eta_2')/2$ and $b = (-b_1'/\eta_1' - b_2'/\eta_2')/2$.

3.2.2 Soft classifier Linear ε-Insensitive TWSVR

In this section, we briefly describe the soft margin classifier ε-insensitive TWSVR for the linear case. Adding an error term in objective function, the soft classifier formulation can be obtained in the following form:

$$\min_{\mathbf{w}_1, b_1, \boldsymbol{\xi}_1} \quad \frac{1}{2}(\mathbf{y}-(A\mathbf{w}_1+\mathbf{e}b_1))^t(\mathbf{y}-(A\mathbf{w}_1+\mathbf{e}b_1))+C_1\mathbf{e}^t\boldsymbol{\xi}_1$$

subject to $\quad \mathbf{y}-(A\mathbf{w}_1+\mathbf{e}b_1) \le \mathbf{e}\varepsilon_1+\boldsymbol{\xi}_1, \quad \boldsymbol{\xi}_1 \ge \mathbf{0}$ (3.7)

$$\min_{\mathbf{w}_2, b_2, \boldsymbol{\xi}_2} \quad \frac{1}{2}(\mathbf{y}-(A\mathbf{w}_2+\mathbf{e}b_2))^t(\mathbf{y}-(A\mathbf{w}_2+\mathbf{e}b_2))+C_2\mathbf{e}^t\boldsymbol{\xi}_2$$

subject to $\quad (A\mathbf{w}_2+\mathbf{e}b_2)-\mathbf{y} \le \mathbf{e}\varepsilon_2+\boldsymbol{\xi}_2, \quad \boldsymbol{\xi}_2 \ge \mathbf{0},$ (3.8)

where $C_1, C_2 > 0$ are regularization parameters and $\boldsymbol{\xi}_1=(\xi_{11},\cdots,\xi_{1m})^t$, $\boldsymbol{\xi}_2=(\xi_{21},\cdots,\xi_{2m})^t$ are slack vectors.

Introducing Lagrange multipliers $\boldsymbol{\alpha}_1=(\alpha_{11},\cdots,\alpha_{1m})^t, \boldsymbol{\beta}_1=(\beta_{11},\cdots,\beta_{1m})^t$, $\boldsymbol{\alpha}_2=(\alpha_{21},\cdots,\alpha_{2m})^t$ and $\boldsymbol{\beta}_2=(\beta_{21},\cdots,\beta_{2m})^t$ the Lagrangian functions for (3.7) and (3.8) can be written as:

$$\begin{aligned} L_1(\mathbf{w}_1, b_1, \boldsymbol{\xi}_1) = & \frac{1}{2}(\mathbf{y}-(A\mathbf{w}_1+\mathbf{e}b_1))^t(\mathbf{y}-(A\mathbf{w}_1+\mathbf{e}b_1))+C_1\mathbf{e}^t\boldsymbol{\xi}_1 \\ & -\boldsymbol{\alpha}_1^t(\mathbf{e}\varepsilon_1+\boldsymbol{\xi}_1-\mathbf{y}+(A\mathbf{w}_1+\mathbf{e}b_1))-\boldsymbol{\beta}_1^t\boldsymbol{\xi}_1 \end{aligned} \tag{3.9}$$

and

$$\begin{aligned} L_2(\mathbf{w}_2, b_2, \boldsymbol{\xi}_2) = & \frac{1}{2}(\mathbf{y}-(A\mathbf{w}_2+\mathbf{e}b_2))^t(\mathbf{y}-(A\mathbf{w}_2+\mathbf{e}b_2))+C_2\mathbf{e}^t\boldsymbol{\xi}_2 \\ & -\boldsymbol{\alpha}_2^t(\mathbf{e}\varepsilon_2+\boldsymbol{\xi}_2-(A\mathbf{w}_2+\mathbf{e}b_2)+\mathbf{y}))-\boldsymbol{\beta}_2^t\boldsymbol{\xi}_2 \end{aligned} \tag{3.10}$$

respectively. Applying the KKT conditions for Lagrangian function (3.9), we get:

$$-\mathbf{A}^t(\mathbf{y}-\mathbf{A}\mathbf{w}_1-\mathbf{e}b_1)-\mathbf{A}^t\boldsymbol{\alpha}=\mathbf{0} \tag{3.11}$$

$$-\mathbf{e}^t(\mathbf{y}-\mathbf{A}\mathbf{w}_1-\mathbf{e}b_1)-\mathbf{e}^t\boldsymbol{\alpha}_1=\mathbf{0} \tag{3.12}$$

$$C_1\mathbf{e}-\boldsymbol{\alpha}_1-\boldsymbol{\beta}_1=\mathbf{0} \tag{3.13}$$

$$\mathbf{y}-\mathbf{A}\mathbf{w}_1-\mathbf{e}b_1 \le \mathbf{e}\varepsilon_1+\boldsymbol{\xi}_1, \quad \boldsymbol{\xi}_1 \ge \mathbf{0} \tag{3.14}$$

$$\boldsymbol{\alpha}_1^t(\mathbf{e}\varepsilon_1+\boldsymbol{\xi}_1-(\mathbf{y}-\mathbf{A}\mathbf{w}_1-\mathbf{e}b_1))=0, \quad \boldsymbol{\alpha}_1 \ge \mathbf{0} \tag{3.15}$$

$$\boldsymbol{\beta}_1^t\boldsymbol{\xi}_1=0, \quad \boldsymbol{\beta}_1 \ge \mathbf{0}. \tag{3.16}$$

Since $\boldsymbol{\beta}_1 \geq \mathbf{0}$, from (3.13) we get

$$\mathbf{0} \leq \boldsymbol{\alpha}_1 \leq C_1\mathbf{e}. \tag{3.17}$$

Similarly for the Lagrangian function (3.10), we get

$$-\mathbf{A}^t(\mathbf{y}-\mathbf{A}\mathbf{w}_2-\mathbf{e}b_2)+\mathbf{A}^t\boldsymbol{\alpha}_2=\mathbf{0} \tag{3.18}$$

$$-\mathbf{e}^t(\mathbf{y}-\mathbf{A}\mathbf{w}_2-\mathbf{e}b_2)+\mathbf{e}^t\boldsymbol{\alpha}_2=0 \tag{3.19}$$

$$C_2\mathbf{e}-\boldsymbol{\alpha}_2-\boldsymbol{\beta}_2=\mathbf{0} \tag{3.20}$$

$$(\mathbf{A}\mathbf{w}_2+\mathbf{e}b_2)-\mathbf{y}\leq \mathbf{e}\varepsilon_2+\boldsymbol{\xi}_2, \quad \boldsymbol{\xi}_2\geq \mathbf{0} \tag{3.21}$$

$$\boldsymbol{\alpha}_2^t(\mathbf{e}\varepsilon_2+\boldsymbol{\xi}_2-(\mathbf{A}\mathbf{w}_2+\mathbf{e}b_2-\mathbf{y})=0, \quad \boldsymbol{\alpha}_2\geq\mathbf{0} \tag{3.22}$$

$$\boldsymbol{\beta}_2^t\boldsymbol{\xi}_2=0, \quad \boldsymbol{\beta}_2\geq\mathbf{0}. \tag{3.23}$$

Since $\boldsymbol{\beta}_2 \geq 0$, from (3.20) we get

$$\mathbf{0}\leq\boldsymbol{\alpha}_2\leq C_2\mathbf{e}. \tag{3.24}$$

Now, combining (3.11) with (3.12) and (3.18) with (3.19), we get

$$-\begin{bmatrix}\mathbf{A}^t\\ \mathbf{e}^t\end{bmatrix}\left(\mathbf{y}-[\mathbf{A}\quad\mathbf{e}]\begin{bmatrix}\mathbf{w}_1\\ b_1\end{bmatrix}\right)-\begin{bmatrix}\mathbf{A}^t\\ \mathbf{e}^t\end{bmatrix}\boldsymbol{\alpha}_1=\mathbf{0} \tag{3.25}$$

$$-\begin{bmatrix}\mathbf{A}^t\\ \mathbf{e}^t\end{bmatrix}\left(\mathbf{y}-[\mathbf{A}\quad\mathbf{e}]\begin{bmatrix}\mathbf{w}_2\\ b_2\end{bmatrix}\right)+\begin{bmatrix}\mathbf{A}^t\\ \mathbf{e}^t\end{bmatrix}\boldsymbol{\alpha}_2=\mathbf{0} \tag{3.26}$$

Equation (3.25) and (3.26) can also be transformed into equations (3.27) and (3.28) respectively, i.e.

$$-G^t\mathbf{f}+G^tG\mathbf{u}_1-G^t\boldsymbol{\alpha}_1=\mathbf{0} \qquad \text{i.e. } \mathbf{u}_1=(G^tG)^{-1}G^t(\mathbf{f}+\boldsymbol{\alpha}_1), \tag{3.27}$$

$$-G^t\mathbf{f}+G^tG\mathbf{u}_2+G^t\boldsymbol{\alpha}_2=\mathbf{0} \qquad \text{i.e. } \mathbf{u}_2=(G^tG)^{-1}G^t(\mathbf{f}-\boldsymbol{\alpha}_2), \tag{3.28}$$

where $G=[\mathbf{A}\quad\mathbf{e}]$, $\mathbf{f}=\mathbf{y}$, $\mathbf{u}_1=[\mathbf{w}_1^t\quad b_1]^t$, and $\mathbf{u}_2=[\mathbf{w}_2^t\quad b_2]^t$. Again to overcome the situation in which the inverse of G^tG does not exist, a regularization term σI can be introduced so that $(G^tG+\sigma I)$ becomes positive definite with σ being a very small positive number, such as $\sigma=1e-7$.

Substituting (3.27) in (3.9) and using (3.14) to (3.17), the dual of (3.7) can be obtained as

$$\min_{\boldsymbol{\alpha}_1 \in R^m} \quad \frac{1}{2}\boldsymbol{\alpha}_1^t G(G^tG)^{-1}G^t\boldsymbol{\alpha}_1 + \mathbf{f}^t G(G^tG)^{-1}G^t\boldsymbol{\alpha}_1 - \mathbf{f}^t\boldsymbol{\alpha}_1 - \varepsilon_1\mathbf{e}^t\boldsymbol{\alpha}_1$$

subject to $\quad \mathbf{0} \le \boldsymbol{\alpha}_1 \le C_1\mathbf{e}.$ (3.29)

Similarly, substituting (3.28) in (3.10) and using (3.21) to (3.24), the dual of (3.8) can be obtained as

$$\min_{\boldsymbol{\alpha}_2 \in R^m} \quad \frac{1}{2}\boldsymbol{\alpha}_2^t G(G^tG)^{-1}G^t\boldsymbol{\alpha}_2 - \mathbf{f}^t G(G^tG)^{-1}G^t\boldsymbol{\alpha}_2 + \mathbf{f}^t\boldsymbol{\alpha}_2 + \varepsilon_2\mathbf{e}^t\boldsymbol{\alpha}_2$$

subject to $\quad \mathbf{0} \le \boldsymbol{\alpha}_2 \le C_2\mathbf{e}.$ (3.30)

The above formulations (3.29) and (3.30) determine hyperplanes $f_1(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + b_1$ and $f_2(\mathbf{x}) = \mathbf{w}_2^t\mathbf{x} + b_2$.The end regressor is obtained by taking the mean of $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$.

3.3 Nonlinear TWSVR

For the nonlinear case, TWSVR determines the ε-insensitive down and up bound functions to be

$$f_1(\mathbf{x}) = K(\mathbf{x}^t, A^t)\mathbf{w}_1 + b_1, \tag{3.31}$$

$$f_2(\mathbf{x}) = K(\mathbf{x}^t, A^t)\mathbf{w}_2 + b_2, \tag{3.32}$$

These hyperplanes are determined by the TWSVR as the solution of the following pair of QPPs:

$$\min_{(\mathbf{w}_1, b_1, \boldsymbol{\xi}_1) \in R^{(n+1+m)}} \frac{1}{2}(\mathbf{y} - (K(\mathbf{A}, \mathbf{A}^t)\mathbf{w}_1 + \mathbf{e}b_1))^t(\mathbf{y} - (K(\mathbf{A}, \mathbf{A}^t)\mathbf{w}_1 + \mathbf{e}b_1)) + C_1\mathbf{e}^t\boldsymbol{\xi}_1$$

subject to $\quad \mathbf{y} - (K(A, A^t)\mathbf{w}_1 + \mathbf{e}b_1) \le \mathbf{e}\varepsilon_1 + \boldsymbol{\xi}_1, \quad \boldsymbol{\xi}_1 \ge \mathbf{0}$ (3.33)

$$\min_{(\mathbf{w}_2, b_2, \boldsymbol{\xi}_2) \in R^{(n+1+m)}} \frac{1}{2}(\mathbf{y} - (K(A, A^t)\mathbf{w}_2 + \mathbf{e}b_2))^t(\mathbf{y} - (K(A, A^t)\mathbf{w}_2 + \mathbf{e}b_2)) + C_2\mathbf{e}^t\boldsymbol{\xi}_2$$

subject to $\quad (K(A, A^t)\mathbf{w}_2 + \mathbf{e}b_2) - \mathbf{y} \le \mathbf{e}\varepsilon_2 + \boldsymbol{\xi}_2, \quad \boldsymbol{\xi}_2 \ge \mathbf{0}$ (3.34)

Introducing Lagrange multipliers $\boldsymbol{\alpha}_1 = (\alpha_{11}, \cdots, \alpha_{1m})^t, \boldsymbol{\beta}_1 = (\beta_{11}, \cdots, \beta_{1m})^t$, $\boldsymbol{\alpha}_2 = (\alpha_{21}, \cdots, \alpha_{2m})^t$ and $\boldsymbol{\beta}_2 = (\beta_{21}, \cdots, \beta_{2m})^t$ their Lagrangian functions can be written as:

$$\begin{aligned} L_1(\mathbf{w}_1, b_1, \boldsymbol{\xi}_1) = \frac{1}{2}(\mathbf{y} - (K(\mathbf{A}, \mathbf{A}^t)\mathbf{w}_1 + \mathbf{e}b_1))^t(\mathbf{y} - (K(\mathbf{A}, \mathbf{A}^t)\mathbf{w}_1 + \mathbf{e}b_1)) \\ + C_1\mathbf{e}^t\boldsymbol{\xi}_1 - \boldsymbol{\alpha}_1^t(\mathbf{e}\varepsilon_1 + \boldsymbol{\xi}_1 - \mathbf{y} + (K(\mathbf{A}, \mathbf{A}^t)\mathbf{w}_1 + \mathbf{e}b_1)) - \boldsymbol{\beta}_1^t\boldsymbol{\xi}_1 \end{aligned} \tag{3.35}$$

$$\begin{aligned} L_2(\mathbf{w}_2, b_2, \boldsymbol{\xi}_2) = \frac{1}{2}(\mathbf{y} - (K(\mathbf{A}, \mathbf{A}^t)\mathbf{w}_2 + \mathbf{e}b_2))^t(\mathbf{y} - (K(\mathbf{A}, \mathbf{A}^t)\mathbf{w}_2 + \mathbf{e}b_2)) \\ + C_2\mathbf{e}^t\boldsymbol{\xi}_2 - \boldsymbol{\alpha}_2^t(\mathbf{e}\varepsilon_2 + \boldsymbol{\xi}_2 - (K(\mathbf{A}, \mathbf{A}^t)\mathbf{w}_2 + \mathbf{e}b_2) + y) - \boldsymbol{\beta}_2^t\boldsymbol{\xi}_2 \end{aligned} \tag{3.36}$$

Proceeding as we have done for the linear case, the duals of (3.33) and (3.34) can be obtained in the following form:

$$\min_{\boldsymbol{\alpha}_1 \in R^m} \quad \frac{1}{2}\boldsymbol{\alpha}_1^t G(G^tG)^{-1}G^t\boldsymbol{\alpha}_1 + \mathbf{f}^tG(G^tG)^{-1}G^t\boldsymbol{\alpha}_1 - \mathbf{f}^t\boldsymbol{\alpha}_1 + \varepsilon_1\mathbf{e}^t\boldsymbol{\alpha}_1$$

subject to $\quad \mathbf{0} \le \boldsymbol{\alpha}_1 \le C_1\mathbf{e}$; (3.37)

$$\min_{\boldsymbol{\alpha}_2 \in R^m} \quad \frac{1}{2}\boldsymbol{\alpha}_2^t G(G^tG)^{-1}G^t\boldsymbol{\alpha}_2 - \mathbf{f}^tG(G^tG)^{-1}G^t\boldsymbol{\alpha}_2 + \mathbf{f}^t\boldsymbol{\alpha}_2 + \varepsilon_2\mathbf{e}^t\boldsymbol{\alpha}_2$$

subject to $\quad \mathbf{0} \le \boldsymbol{\alpha}_2 \le C_2\mathbf{e}$. (3.38)

$$\mathbf{u}_1 = (G^tG + \sigma I)^{-1}G^t(\mathbf{f} - \boldsymbol{\alpha}_1), \tag{3.39}$$

$$\mathbf{u}_2 = (G^tG + \sigma I)^{-1}G^t(\mathbf{f} + \boldsymbol{\alpha}_2), \tag{3.40}$$

where $\mathbf{u}_1 = [\mathbf{w}_1^t \quad b_1]^t$, $\mathbf{u}_2 = [\mathbf{w}_2^t \quad b_2]^t$ and $G = [K(A, A^t) \quad \mathbf{e}]$.

For a new data point $\mathbf{x} \in R^n$, the end regressor can be obtained as

$$f(\mathbf{x}) = \frac{1}{2}[K(\mathbf{x}^t, A^t) \quad 1]^t(\mathbf{u}_1 + \mathbf{u}_2)$$

Chapter 4

LTWSVR: Lagrangian Twin Support Vector Regression Based on Twin Support Vector Machine

4.1 Introduction

In this work, an implicit Lagrangian for the 2-norm TWSVR is proposed. This Lagrangian formulation is motivated from the study of (Mangasarian and Musicant, 2001) for classification problem as an unconstrained differentiable convex problem. Further it is proposed to solve this problem by a simple and linearly convergent iterative Lagrangian twin support vector regression method based on twin SVM (LTWSVR) algorithm. LTWSVR requires at the outset the inverse of a matrix but this can be expressed as matrix subtraction of identity matrix by a scalar multiple of the inverse of a positive semi-definite matrix (Balasundaram and Tanveer, 2013). LTWSVR does not need any optimization tools of linear or quadratic programming solvers.

Inspired by the study of Finite Newton method for Lagrangian SVM for Classification proposed in (Fung & Mangasarian, 2003) , Newton method for implicit Lagrangian formulation is discussed i.e. unconstrained minimization problems corresponding to the duals of TWSVR is also proposed in section 4.3.

The chapter is organized as follows. In section 4.2 we derive the linear and nonlinear Lagrangian TWSVR (LTWSVR) by formulating the TWSVR in 2-norm as an unconstrained minimization problem (Balasundaram and Tanveer, 2013) and obtain its dual. In section 4.3 we describe Newton method for solving this unconstrained minimization problem. In section 4.4 we consider LTWSVR as an absolute value equation problem and it is proposed to obtain solution using Newton method. We also propose a generalized derivative approach based solution in section 4.5.

4.2 Lagrangian Twin Support Vector Regression Based on Twin Support Vector Machine (LTWSVR)

For the linear TWSVR in 2-norm, it's up-bound $f_1(.)$ and down-bound $f_2(.)$ regressor of the form (2.2) and (2.3) are determined by solving the pair of QPPs:

$$\min_{\mathbf{w}_1, b_1, \boldsymbol{\xi}_1} \ \frac{1}{2}\left(\mathbf{y} - [A \quad \mathbf{e}]\begin{bmatrix}\mathbf{w}_1 \\ b_1\end{bmatrix}\right)^t \left(\mathbf{y} - [A \quad \mathbf{e}]\begin{bmatrix}\mathbf{w}_1 \\ b_1\end{bmatrix}\right) + \frac{C_1}{2}\boldsymbol{\xi}_1^t\boldsymbol{\xi}_1$$

subject to $$\left(\mathbf{y} - [A \quad \mathbf{e}]\begin{bmatrix}\mathbf{w}_1 \\ b_1\end{bmatrix}\right) \le \mathbf{e}\varepsilon_1 + \boldsymbol{\xi}_1 \tag{4.1}$$

and

$$\min_{\mathbf{w}_2, b_2, \boldsymbol{\xi}_2} \ \frac{1}{2}\left(\mathbf{y} - [A \quad \mathbf{e}]\begin{bmatrix}\mathbf{w}_2 \\ b_2\end{bmatrix}\right)^t \left(\mathbf{y} - [A \quad \mathbf{e}]\begin{bmatrix}\mathbf{w}_2 \\ b_2\end{bmatrix}\right) + \frac{C_2}{2}\boldsymbol{\xi}_2^t\boldsymbol{\xi}_2$$

subject to $$\left([A \quad \mathbf{e}]\begin{bmatrix}\mathbf{w}_2 \\ b_2\end{bmatrix} - \mathbf{y}\right) \le \mathbf{e}\varepsilon_2 + \boldsymbol{\xi}_2 \tag{4.2}$$

where $C_1, C_2 > 0$ and slack vectors $\boldsymbol{\xi}_1 = (\xi_{11}, \cdots, \xi_{1m})^t$, $\boldsymbol{\xi}_2 = (\xi_{21}, \cdots, \xi_{2m})^t$. Note that the non-negative constraints of $\boldsymbol{\xi}_1$ and $\boldsymbol{\xi}_2$ have been dropped in (4.1) and (4.2) because they will be satisfied automatically at optimality.

Let $G = [A \quad \mathbf{e}]_{m\times(n+1)}$. Then (4.1) and (4.2) become

$$\min_{\mathbf{w}_1, b_1, \boldsymbol{\xi}_1} \ \frac{1}{2}\left(\mathbf{y} - G\begin{bmatrix}\mathbf{w}_1 \\ b_1\end{bmatrix}\right)^t \left(\mathbf{y} - G\begin{bmatrix}\mathbf{w}_1 \\ b_1\end{bmatrix}\right) + \frac{C_1}{2}\boldsymbol{\xi}_1^t\boldsymbol{\xi}_1$$

subject to $$\left(\mathbf{y} - G\begin{bmatrix}\mathbf{w}_1 \\ b_1\end{bmatrix}\right) \le \mathbf{e}\varepsilon_1 + \boldsymbol{\xi}_1 \tag{4.3}$$

and

$$\min_{\mathbf{w}_2, b_2, \boldsymbol{\xi}_2} \ \frac{1}{2}\left(\mathbf{y} - G\begin{bmatrix}\mathbf{w}_2 \\ b_2\end{bmatrix}\right)^t \left(\mathbf{y} - G\begin{bmatrix}\mathbf{w}_2 \\ b_2\end{bmatrix}\right) + \frac{C_2}{2}\boldsymbol{\xi}_2^t\boldsymbol{\xi}_2$$

subject to $$\left(G\begin{bmatrix}\mathbf{w}_2 \\ b_2\end{bmatrix} - \mathbf{y}\right) \le \mathbf{e}\varepsilon_2 + \boldsymbol{\xi}_2 \tag{4.4}$$

Introducing Lagrange multiplier $\mathbf{u}_1 = (u_{11}, \cdots, u_{1m})^t$, the Lagrangian function $L_1(.)$ corresponding to (4.3) can be written as:

$$L_1(\mathbf{w}_1, b_1, \xi_1) = \frac{1}{2}\left[\left(\mathbf{y} - G\begin{bmatrix}\mathbf{w}_1\\ b_1\end{bmatrix}\right)^t\left(\mathbf{y} - G\begin{bmatrix}\mathbf{w}_1\\ b_1\end{bmatrix}\right) + C_1\xi_1^t\xi_1\right] + \mathbf{u}_1^t\left[\mathbf{y} - G\begin{bmatrix}\mathbf{w}_1\\ b_1\end{bmatrix} - \mathbf{e}\varepsilon_1 - \xi_1\right]$$

Applying the KKT conditions for Lagarangian function $L_1(.)$, we get:

$$\frac{\partial L_1}{\partial\begin{bmatrix}\mathbf{w}_1\\ b_1\end{bmatrix}} = 0 \quad \Rightarrow -G^t\left(y - G\begin{bmatrix}\mathbf{w}_1\\ b_1\end{bmatrix}\right) - G^t\mathbf{u}_1 = 0 \quad \Rightarrow \begin{bmatrix}\mathbf{w}_1\\ b_1\end{bmatrix} = (G^tG)^{-1}G^t(\mathbf{y} + \mathbf{u}_1) \quad \text{and}$$

$$\frac{\partial L_1}{\partial \xi_1} = 0 \quad \Rightarrow C_1\xi_1 - \mathbf{u}_1 = 0 \quad \Rightarrow \xi_1 = \frac{\mathbf{u}_1}{C_1}.$$

Substituting these results back into the Lagarangian function $L_1(.)$, and ignoring the constant terms the dual of (4.3) can be written as a minimization problem of the form

$$\min_{\mathbf{u}_1 \geq \mathbf{0}} \quad L_1(\mathbf{u}_1) = \frac{1}{2}\mathbf{u}_1^t\left(\frac{I}{C_1} + H\right)\mathbf{u}_1 - (\mathbf{y}^t - \mathbf{y}^tH - \mathbf{e}^t\varepsilon_1)\mathbf{u}_1 \tag{4.5}$$

where $H = G(G^tG)^{-1}G^t$.

Similarly by introducing the Lagrange multiplier $\mathbf{u}_2 = (u_{21}, \cdots, u_{2m})^t$, $L_2(.)$ can be written as:

$$L_2(\mathbf{w}_2, b_2, \xi_2) = \frac{1}{2}\left[\left(\mathbf{y} - G\begin{bmatrix}\mathbf{w}_2\\ b_2\end{bmatrix}\right)^t\left(\mathbf{y} - G\begin{bmatrix}\mathbf{w}_2\\ b_2\end{bmatrix}\right) + C_2\xi_2^t\xi_2\right] + \mathbf{u}_2^t\left[G\begin{bmatrix}\mathbf{w}_2\\ b_2\end{bmatrix} - \mathbf{y} - \varepsilon_2\mathbf{e} - \xi_2\right]$$

Applying the KKT conditions for Lagrange function $L_2(.)$, we get:

$$\frac{\partial L_2}{\partial\begin{bmatrix}\mathbf{w}_2\\ b_2\end{bmatrix}} = 0 \quad \Rightarrow -G^t\left(y - G\begin{bmatrix}\mathbf{w}_2\\ b_2\end{bmatrix}\right) + G^t\mathbf{u}_2 = 0 \quad \Rightarrow \begin{bmatrix}\mathbf{w}_2\\ b_2\end{bmatrix} = (G^tG)^{-1}G^t(\mathbf{y} - \mathbf{u}_2) \quad \text{and}$$

$$\frac{\partial L_2}{\partial \xi_2} = 0 \quad \Rightarrow C_2\xi_2 - \mathbf{u}_2 = 0 \quad \Rightarrow \xi_2 = \frac{\mathbf{u}_2}{C_2}.$$

Substituting these results back into Lagrangian function $L_2(.)$, we get:

$$\min_{\mathbf{u}_2 \geq \mathbf{0}} \quad L_2(\mathbf{u}_2) = \frac{1}{2}\mathbf{u}_2^t\left(\frac{I}{C_2} + H\right)\mathbf{u}_2 - (\mathbf{y}^t H - \mathbf{y}^t - \mathbf{e}^t \varepsilon_2)\mathbf{u}_2 \tag{4.6}$$

The above minimization problems (4.5) and (4.6) can be equivalently written in the following simpler form:

$$\min_{\mathbf{u}_1 \geq \mathbf{0}} \quad L_1(\mathbf{u}_1) = \frac{1}{2}\mathbf{u}_1^t Q_1 \mathbf{u}_1 - \mathbf{r}_1^t \mathbf{u}_1 \quad \text{and}$$

$$\min_{\mathbf{u}_2 \geq \mathbf{0}} \quad L_2(\mathbf{u}_2) = \frac{1}{2}\mathbf{u}_2^t Q_2 \mathbf{u}_2 - \mathbf{r}_2^t \mathbf{u}_2 \tag{4.7}$$

respectively, where

$Q_1 = \frac{I}{C_1} + H, Q_2 = \frac{I}{C_2} + H, r_1 = (I - H)\mathbf{y} - \varepsilon_1 \mathbf{e}$ and $r_1 = (H - I)\mathbf{y} - \varepsilon_2 \mathbf{e}$.

Each of the above two QPPs determines the functions

$$f_1(x) = [\mathbf{x}^t \quad 1]\begin{bmatrix}\mathbf{w}_1 \\ b_1\end{bmatrix} = [\mathbf{x}^t \quad 1](G^t G)^{-1} G^t(\mathbf{y} + \mathbf{u}_1) \quad \text{and}$$

$$f_2(x) = [\mathbf{x}^t \quad 1]\begin{bmatrix}\mathbf{w}_2 \\ b_2\end{bmatrix} = [\mathbf{x}^t \quad 1](G^t G)^{-1} G^t(\mathbf{y} - \mathbf{u}_2) \tag{4.8}$$

For the nonlinear TWSVR in 2-norm, it's up-bound $f_1(.)$ and down-bound $f_2(.)$ regressors are determined by solving the pair of QPPs:

$$\min_{\mathbf{w}_1, b_1, \xi_1} \quad \frac{1}{2}\left(\mathbf{y} - [K(A, A^t) \quad \mathbf{e}]\begin{bmatrix}\mathbf{w}_1 \\ b_1\end{bmatrix}\right)^t \left(\mathbf{y} - [A \quad \mathbf{e}]\begin{bmatrix}\mathbf{w}_1 \\ b_1\end{bmatrix}\right) + \frac{C_1}{2}\xi_1^t \xi_1$$

$$\text{s.t.} \qquad \left(\mathbf{y} - [K(A, A^t) \quad \mathbf{e}]\begin{bmatrix}\mathbf{w}_1 \\ b_1\end{bmatrix}\right) \leq \mathbf{e}\varepsilon_1 + \xi_1 \tag{4.9}$$

$$\text{and} \quad \min_{\mathbf{w}_2, b_2, \xi_2} \quad \frac{1}{2}\left(\mathbf{y} - [K(A, A^t) \quad \mathbf{e}]\begin{bmatrix}\mathbf{w}_2 \\ b_2\end{bmatrix}\right)^t \left(\mathbf{y} - [A \quad \mathbf{e}]\begin{bmatrix}\mathbf{w}_2 \\ b_2\end{bmatrix}\right) + \frac{C_2}{2}\xi_2^t \xi_2$$

$$\text{s.t.} \qquad \left([K(A, A^t) \quad \mathbf{e}]\begin{bmatrix}\mathbf{w}_2 \\ b_2\end{bmatrix} - \mathbf{y}\right) \leq \mathbf{e}\varepsilon_2 + \xi_2 \tag{4.10}$$

Proceeding as we have done for the linear TWSVR, the dual QPPs of (4.9) and (4.10) can be obtained as a pair of minimization problem of the form (4.7) where $\mathbf{u}_1, \mathbf{u}_2, Q_1, Q_2, \mathbf{r}_1, \mathbf{r}_2$ has the same definition as defined above but the augmented matrix G is defined by: $G = [K(A, A^t) \quad \mathbf{e}]_{m\times(m+1)}$.The kernel regressor functions can be determined as the mean of the up-bound $f_1(.)$ and down-bound $f_2(.)$ regressor functions as follows:

$$f_1(x) = [K(\mathbf{x}^t, A^t) \quad 1]\begin{bmatrix}\mathbf{w}_1 \\ b_1\end{bmatrix} = [K(\mathbf{x}^t, A^t) \quad 1](G^tG)^{-1}G^t(\mathbf{y}+\mathbf{u}_1) \quad \text{and}$$

$$f_2(x) = [K(\mathbf{x}^t, A^t) \quad 1]\begin{bmatrix}\mathbf{w}_2 \\ b_2\end{bmatrix} = [K(\mathbf{x}^t, A^t) \quad 1](G^tG)^{-1}G^t(\mathbf{y}-\mathbf{u}_2) \tag{4.11}$$

Now we discuss the solution of dual QPPs (4.7) by our iterative LTWSVR algorithm.

The KKT necessary and sufficient optimal conditions (Mangasarian, 1994) for the dual QPPs (4.7) will become

$$\mathbf{0} \le \mathbf{u}_1 \perp (Q_1\mathbf{u}_1 - \mathbf{r}_1) \ge \mathbf{0} \quad \text{and} \quad \mathbf{0} \le \mathbf{u}_2 \perp (Q_2\mathbf{u}_2 - \mathbf{r}_2) \ge \mathbf{0} \tag{4.12}$$

For any two vectors $\mathbf{a}$ and $\mathbf{b}$ the following identity holds

$$\mathbf{0} \le \mathbf{a} \perp \mathbf{b} \ge \mathbf{0} \Leftrightarrow \mathbf{a} = (\mathbf{a} - \alpha\mathbf{b})_+, \quad \alpha > 0$$

Using this identity optimal condition, for any $\alpha_1, \alpha_2 > 0$ (4.12) can be written as

$$Q_1\mathbf{u}_1 - \mathbf{r}_1 = ((Q_1\mathbf{u}_1 - \mathbf{r}_1) - \alpha_1\mathbf{u}_1)_+ \quad \text{and} \quad Q_2\mathbf{u}_2 - \mathbf{r}_2 = ((Q_2\mathbf{u}_2 - \mathbf{r}_2) - \alpha_2\mathbf{u}_2)_+ \tag{4.13}$$

These optimality conditions are also the necessary and sufficient condition for the unconstrained minimum of the implicit Lagrangian (Mangasarian and Solodov, 1993) associated with the dual problems (4.7):

$$\min_{\mathbf{u}_1 \in R^m} \quad L_1(\mathbf{u}_1) = \frac{1}{2}\mathbf{u}_1^tQ_1\mathbf{u}_1 - \mathbf{r}_1^t\mathbf{u}_1 + \frac{1}{2\alpha_1}\left(\|(-\alpha_1\mathbf{u}_1 + Q_1\mathbf{u}_1 - \mathbf{r}_1)_+\|^2 - \|Q_1\mathbf{u}_1 - \mathbf{r}_1\|^2\right)$$

and

$$\min_{\mathbf{u}_2 \in R^m} \quad L_2(\mathbf{u}_2) = \frac{1}{2}\mathbf{u}_2^tQ_2\mathbf{u}_2 - \mathbf{r}_2^t\mathbf{u}_2 + \frac{1}{2\alpha_2}\left(\|(-\alpha_2\mathbf{u}_2 + Q_2\mathbf{u}_2 - \mathbf{r}_2)_+\|^2 - \|Q_2\mathbf{u}_2 - \mathbf{r}_2\|^2\right)$$

(4.14)

The optimality conditions (4.12) can also be written as

$$\mathbf{0} \le \frac{\mathbf{u}_1}{C_1} \perp \frac{\mathbf{u}_1}{C_1} - (\mathbf{r}_1 - H\mathbf{u}_1) \ge \mathbf{0} \quad \text{and} \quad \mathbf{0} \le \frac{\mathbf{u}_2}{C_2} \perp \frac{\mathbf{u}_2}{C_2} - (\mathbf{r}_2 - H\mathbf{u}_2) \ge \mathbf{0}$$

$$\Leftrightarrow \quad \mathbf{0} \le \frac{\mathbf{u}_1}{C_1} \perp \frac{\mathbf{u}_1}{C_1} - (\mathbf{r}_1 - H\mathbf{u}_1) \ge \mathbf{0} \quad \text{and} \quad \mathbf{0} \le \frac{\mathbf{u}_2}{C_2} \perp \frac{\mathbf{u}_2}{C_2} - (\mathbf{r}_2 - H\mathbf{u}_2) \ge \mathbf{0}$$

$$\Leftrightarrow \quad \frac{\mathbf{u}_1}{C_1} = (\mathbf{r}_1 - H\mathbf{u}_1)_+ \quad \text{and} \quad \frac{\mathbf{u}_2}{C_2} = (\mathbf{r}_2 - H\mathbf{u}_2)_+$$

$$\Leftrightarrow \quad \left(\frac{I}{C_1/2} + H\right)\mathbf{u}_1 = \mathbf{r}_1 + |\mathbf{r}_1 - H\mathbf{u}_1| \quad \text{and} \quad \left(\frac{I}{C_2/2} + H\right)\mathbf{u}_2 = \mathbf{r}_2 + |\mathbf{r}_2 - H\mathbf{u}_2|$$

Finally, we have following simple iterative scheme for LTWSVR algorithm: $i = 0,1,2,\ldots$

$$\mathbf{u}_1 = Q_2^{-1}\left(\mathbf{r}_1 + |\mathbf{r}_1 - H\mathbf{u}_1^i|\right) \quad \text{and} \quad \mathbf{u}_2 = Q_3^{-1}\left(\mathbf{r}_2 + |\mathbf{r}_2 - H\mathbf{u}_2^i|\right) \qquad (4.15)$$

where,

$$\mathbf{r}_1 = (I - H)\mathbf{y} - \varepsilon_1\mathbf{e}, \quad \mathbf{r}_2 = (H - I)\mathbf{y} - \varepsilon_2\mathbf{e}, \quad H = G(G^tG)^{-1}G^t, \quad Q_2 = \left(\frac{I}{C_1/2} + H\right),$$

$$Q_3 = \left(\frac{I}{C_2/2} + H\right) \quad \text{and} \quad G = \begin{cases} [A \quad \mathbf{e}] & \text{(linear)} \\ [K(A, A^t) \quad \mathbf{e}] & \text{(nonlinear)} \end{cases}$$

Remark: The proposed LTWSVR algorithm requires at its very beginning the inverse of matrices Q_2 and Q_3. but this explicit computation are not required because once the matrix $(G^tG)^{-1}$ is known, they can be easily obtained from the result (Balasundaram and Tanveer, 2013)

$$Q_{k+1}^{-1} = C_k\left(I - \frac{C_k/2}{1 + (C_k/2)}H\right) \quad \text{for } k = 1,2.$$

Finally, the end regressor function is defined as:

$$f(\mathbf{x}) = \frac{f_1(x) + f_2(x)}{2}$$

where, for linear case

$$f_1(x) = [\mathbf{x}^t \quad 1](G^tG)^{-1}G^t(\mathbf{y} + \mathbf{u}_1) \text{ and } f_2(x) = [\mathbf{x}^t \quad 1](G^tG)^{-1}G^t(\mathbf{y} - \mathbf{u}_2)$$

Similarly for the nonlinear case,

$$f_1(x) = [K(\mathbf{x}^t, A^t) \quad 1](G^tG)^{-1}G^t(\mathbf{y} + \mathbf{u}_1) \text{ and } f_2(x) = [K(\mathbf{x}^t, A^t) \quad 1](G^tG)^{-1}G^t(\mathbf{y} - \mathbf{u}_2)$$

4.3 Newton method for LTWSVR

In this section, Newton method is described for the solution of the implicit Lagrangian formulations i.e. unconstrained minimization problems (4.14) that lead to highly effective iterative scheme (Fung et al., 2003). In short, (4.14) can also be written as:

$$\min_{\mathbf{u}_k \in R^m} L_k(\mathbf{u}_k) = \frac{1}{2}\mathbf{u}_k^t Q_k \mathbf{u}_k - \mathbf{r}_k^t \mathbf{u}_k + \frac{1}{2\alpha_k}\left(\left\|(-\alpha_k \mathbf{u}_k + Q_k \mathbf{u}_k - \mathbf{r}_k)_+\right\|^2 - \left\|Q_k \mathbf{u}_k - \mathbf{r}_k\right\|^2\right) \quad k = 1,2$$

The basic Newton step for determining the vector $\mathbf{u}_k^{i+1} \in R^m$ from its previous value $\mathbf{u}_k^i$ can be given by the following iterative formula:

$$\nabla L_k(\mathbf{u}_k^i) + \partial^2 L(\mathbf{u}_k^i)(\mathbf{u}_k^{i+1} - \mathbf{u}_k^i) = \mathbf{0}, \quad \text{for } i = 0,1,\ldots \tag{4.16}$$

The gradient of $L_k(\mathbf{u}_k)$ can be obtained as

$$\nabla L_k(\mathbf{u}_k) = \frac{(\alpha_k I - Q_k)}{\alpha_k}((Q_k \mathbf{u}_k - \mathbf{r}_k) - ((Q_k - \alpha_k I)\mathbf{u}_k - \mathbf{r}_k)_+)$$

The Hessian matrix of second order partial derivative of $L_k(\mathbf{u}_k)$ does not exist because gradient $\nabla(L_k(\mathbf{u}_k))$ is not differentiable. However, it has been shown that a generalized Hessian matrix of $L_k(\mathbf{u}_k)$ exist (Facchinei, 1995; Hiriart-Urruty, Strodiot, & Nguyen, 1984) and is defined as follows:

$$\partial^2 L_k(\mathbf{u}_k) = \frac{(\alpha_k I - Q_k)}{\alpha_k}(Q_k + diag((Q_k - \alpha_k I)\mathbf{u}_k - \mathbf{r}_k)_*(\alpha_k I - Q_k))$$

where $diag(.)_*$ is a diagonal matrix and $(\cdot)_*$ denotes the step function, which is taken here as the subgradient of the plus function $(\cdot)_+$, i.e. the step function $\mathbf{x}_*$ denotes a vector $\mathbf{x}$ with all positive components set to 1 and all nonpositive components of $\mathbf{x}$ set to zero (Mangasarian, 2002).

We note that if Q_k be symmetric positive definite matrix and $\alpha_k > \|Q_k\|$ then both $\nabla L_k(\mathbf{u}_k)$ and $\partial^2 L_k(\mathbf{u}_k)$ containing multiplicative factor $\frac{(\alpha_k I - Q_k)}{\alpha_k}$ will be positive definite. So Newton iteration (4.16) is simplified to:

$$h_k(\mathbf{u}_k^i) + \partial h(\mathbf{u}_k^i)(\mathbf{u}_k^{i+1} - \mathbf{u}_k^i) = \mathbf{0} \quad i = 0,1\ldots \tag{4.17}$$

where,

$$h_k(\mathbf{u}_k^i) = (Q_k\mathbf{u}_k^i - \mathbf{r}_k) - ((Q_k - \alpha_k I)\mathbf{u}_k^i - \mathbf{r}_k)_+ = \left(\frac{\alpha_k I - Q_k}{\alpha_k}\right)^{-1} \nabla L_k(\mathbf{u}_k^i) \quad \text{and}$$

$$\partial h_k(\mathbf{u}_k^i) = Q_k + diag((Q_k - \alpha_k I)\mathbf{u}_k^i - \mathbf{r}_k)_* \,(\alpha_k I - Q_k) = \left(\frac{\alpha_k I - Q_k}{\alpha_k}\right)^{-1} \partial^2 L(\mathbf{u}_k^i)$$

4.4 LTWSVR as an absolute value equation problem by Newton method

Again, consider the absolute value equation problem

$$\left(\frac{I}{C_k/2} + H\right)\mathbf{u}_k = \mathbf{r}_k + |\mathbf{r}_k - H\mathbf{u}_k|, \quad k = 1,2.$$

Let, $\quad g_k(\mathbf{u}_k) = \left(\frac{I}{C_k/2} + H\right)\mathbf{u}_k - \left(\mathbf{r}_k + |\mathbf{r}_k - H\mathbf{u}_k|\right), \quad k = 1,2.$

Then, the generalized Jacobian of $g_k(\cdot)$ can be obtained in the following form

$$\partial g_k(\mathbf{u}_k) = \left(\frac{I}{C_k/2} + H\right) + diag(sign(\mathbf{r}_k - H\mathbf{u}_k))H$$

Then, Newton method becomes

$$\partial g_k(\mathbf{u}_k^i)(\mathbf{u}_k^{i+1} - \mathbf{u}_k^i) = -g_k(\mathbf{u}_k^i) \quad i = 0,1\ldots$$

i.e.

$$\left[\left(\frac{I}{C_k/2} + H\right) + diag(sign(\mathbf{r}_k^i - H\mathbf{u}_k^i))H\right](\mathbf{u}_k^{i+1} - \mathbf{u}_k^i)$$

$$= -\left[\left(\frac{I}{C_k/2} + H\right)\mathbf{u}_k^i - \left(\mathbf{r}_k + |\mathbf{r}_k - H\mathbf{u}_k^i|\right)\right]$$

$$k = 1,2 \ \text{ and } \ i = 0,1,\ldots$$

where,

$$\mathbf{r}_1 = (I - H)\mathbf{y} - \varepsilon_1\mathbf{e}, \quad \mathbf{r}_2 = (H - I)\mathbf{y} - \varepsilon_2\mathbf{e} \quad \text{and} \quad H = G(G^tG)^{-1}G^t$$

4.5 Generalized Newton method for LTWSVR

A generalized derivative approach studied in (Fung and Mangasarian, 2003; Balasundaram and Singh, 2010) is described here for solving the unconstrained minimization problems described in section 4.2:

Consider $g_1(\mathbf{u}_1) = \frac{\mathbf{u}_1}{C_1} - (\mathbf{r}_1 - H\mathbf{u}_1)_+$ and $g_2(\mathbf{u}_2) = \frac{\mathbf{u}_2}{C_2} - (\mathbf{r}_2 - H\mathbf{u}_2)_+$.

The gradients should be zero.

Then, using a generalized derivative, the generalized Jacobians of $g_1(\mathbf{u}_1)$ and $g_2(\mathbf{u}_2)$ can be taken as

$$\partial g_1(\mathbf{u}_1) = \frac{I}{C_1} + diag(sign((\mathbf{r}_1 - H\mathbf{u}_1)_+))H$$

and

$$\partial g(\mathbf{u}_2) = \frac{I}{C_2} + diag(sign((\mathbf{r}_2 - H\mathbf{u}_2)_+))H, \text{ respectively.}$$

Using this, a generalized Newton method for solving $g_k(\mathbf{u}_k) = 0, k = 1,2$ becomes

$$\partial g_k(\mathbf{u}_k^i)(\mathbf{u}_k^{i+1} - \mathbf{u}_k^i) = -g_k(\mathbf{u}_k^i) \quad k = 1,2,\ldots \text{ and } i = 0,1\ldots$$

which lead to the following iterative method:

$$\left[\left(\frac{I}{C_k/2} + H\right) + diag(sign((\mathbf{r}_k - H\mathbf{u}_k)_+))H\right](\mathbf{u}_k^{i+1} - \mathbf{u}_k^i)$$
$$= -\left[\frac{\mathbf{u}_k^i}{C_k} - \left(\mathbf{r}_k - (\mathbf{r}_k - H\mathbf{u}_k^i)_+\right)\right]$$

$$k = 1,2 \text{ and } i = 0,1,\ldots$$

Chapter 5

Experimental Results and Analysis

In this chapter, we investigate the effectiveness and speed of the proposed method LTWSVR, defined by gradient based iterative algorithms: FLTWSVR, NLTWSVR and GLTWSVR, on five synthetic and several well known real world datasets. We focus on the comparison of their results with standard SVR, TSVR and TWSVR in terms of accuracy and learning time.

The chapter is organized as follows: we introduce the specification of experimental environment for all computations in section 5.1. We describe the performance of proposed method LTWSVR on synthetic and real world datasets in sections 5.2 and 5.3 respectively.

5.1 Experimental Specification

All experiments are implemented on a PC running Windows 7 with 3.2 GHz Intel CORE i2 processor, 3 GB RAM with MATLAB 2008a. QPPs involved in SVR, TSVR and TWSVR are solved by Mosek optimization toolbox (available online at http://www. mosek.com) for MATLAB which implements fast interior point based algorithms for convex optimization problems. No optimization tool is required for our proposed method LTWSVR. In order to construct nonlinear regressor, Gaussian kernel with parameter $\mu > 0$ defined by $k(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\mu\|\mathbf{x}_1 - \mathbf{x}_2\|^2)$ is utilized. To compare the robustness of the proposed method, root mean square error (RMSE) is employed and is defined as

$$\text{RMSE}=\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y}_i)^2}$$

where y_i and $\bar{y}_i$ are the observed and predicted value for the i^{th} sample respectively and N is the number of test samples.

To reduce the complexity of the optimal parameter selection procedure for TSVR, TWSVR and proposed LTWSVR, we let $C_1 = C_2$ and $\varepsilon_1 = \varepsilon_2$ as in Peng (2010). Furthermore, we let regularization parameters $C, C_1 = C_2 \in [10^{-5}, \ldots, 10^{5}]$, tolerance parameters $\varepsilon, \varepsilon_1 = \varepsilon_2 \in [10^{-3}, \ldots, 10^{-1}]$ and kernel parameter $\mu \in [2^{-5}, \ldots, 2^{5}]$. These optimal parameters are tuned by performing standard ten-fold cross validation on experimental datasets.

5.2 Illustrations and Experiment on Synthetic Datasets

In this experiment section, we evaluate the performance of proposed LTWSVR algorithms on five synthetic datasets generated by the functions which are defined in Table 5.1. For each function $y = f(\mathbf{x})$, we generated 1000 testing samples $(\mathbf{x}, y)$ using $y = f(\mathbf{x})$ and 200 training samples $(\mathbf{x}, y)$ using $y = f(\mathbf{x}) + \eta$ randomly on the intervals defined in Table 5.1, where η is additive noise. Note that for robust comparison, we contaminated 200 training samples with two different kind of noises: a) uniform distribution over the interval [-0.2, 0.2] and b) Gaussian distribution with mean 0 and standard deviation 0.2. The optimal values for regularization, error tolerance and kernel parameters are obtained from their appropriate ranges as described in section 5.1 by performing ten-fold cross validation on the training set. Using these optimal values and a Gaussian kernel, the RMSE on testing set for methods SVR, TSVR, TWSVR, FLTWSVR, NLTWSVR and GLTWSVR were obtained and summarized in Table 5.2. It can be observed from the Table 5.2 that the proposed LTWSVR achieve the competitive generalization performance with much faster learning speed in comparison to SVR, TSVR and TWSVR.

For evaluating the performance of LTWSVR algorithms, the first example considered is the regression of the function (Riberio, 2002) as defined in Table 5.1, i.e.

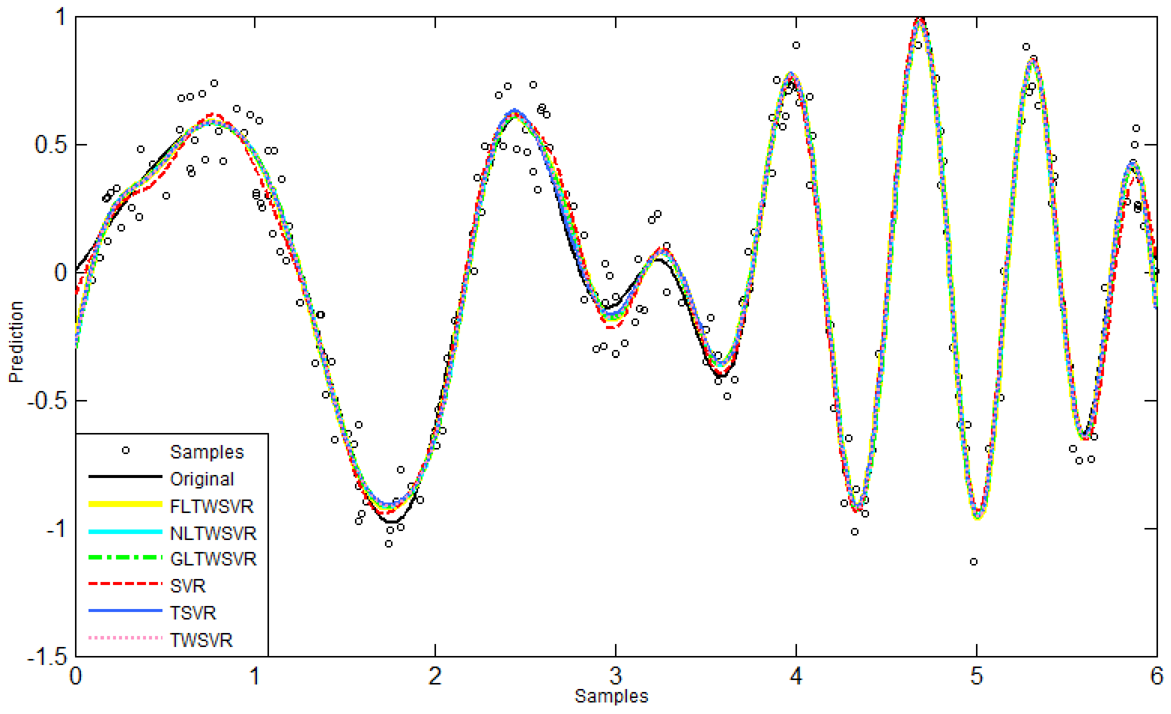
$$f(x) = \sin(x)\cos(x^2), \quad x \in [0, 6]$$

The approximation of this function by SVR, TSVR , TWSVR, FLTWSVR, NLTWSVR and GLTWSVR methods for uniform and Gaussian additive noises over test set were obtained and illustrated in Figures 5.1(a) and 5.1(b) where noisy training samples marked

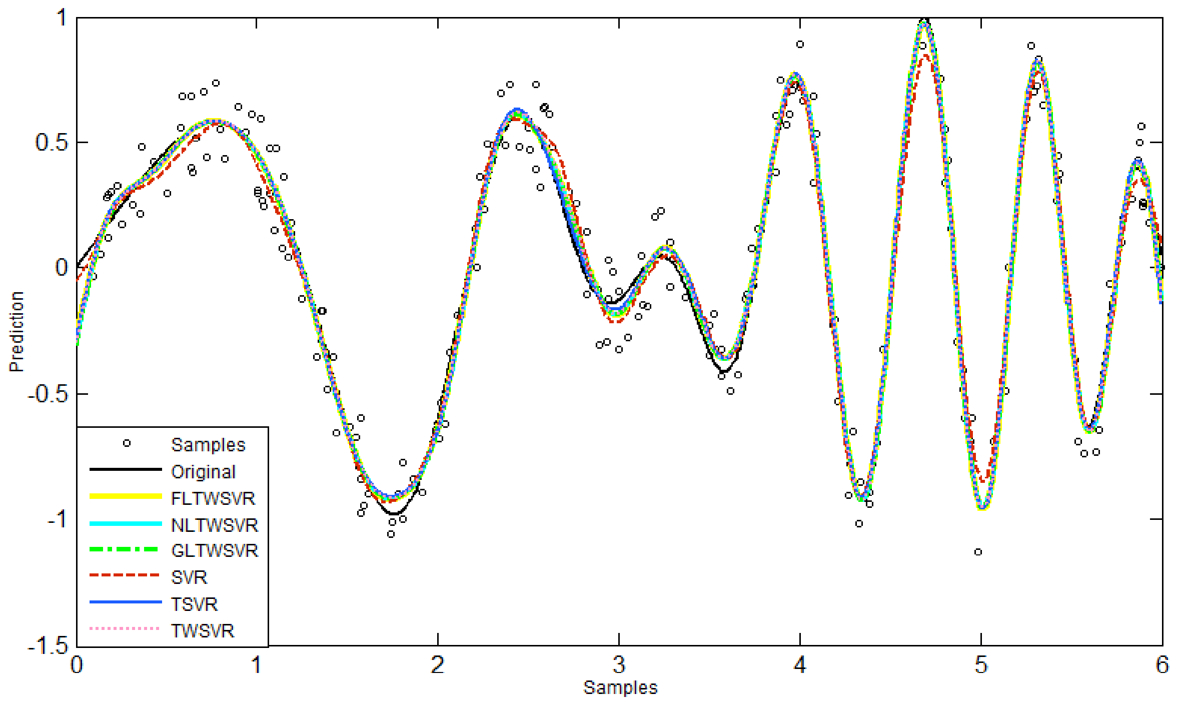
Table 5.1: Functions used for generating synthetic datasets

Name	Function Definition	Domain of Definition
Function 1	$\sin(x)\cos(x^2)$	$x \in [0, 6]$
Function 2	$\exp(x_1 \sin(\pi x_2))$	$x_{i=1,2} \in [-1, 1]$
Function 3	$\sin\left(\frac{2\pi(0.35\times 10+1)}{0.35x+1}\right)$	$x \in [0, 10]$
Function 4	$\frac{(5-x_2)^2}{3(5-x_1)^2+(5-x_2)^2}$	$x_{i=1,2,} \in [0, 10]$
Function 5	$1.9[1.35+\exp(x_1)\sin(13(x_1-0.6)^2) + \exp(3(x_2-0.5))\sin(4\pi(x_2-0.9)^2)]$	$x_{i=1,2,} \in [0, 1]$

by symbol ‘o’. Prediction errors are obtained by taking the difference between the observed and predicted values. The prediction errors by SVR, TSVR, TWSVR, FLTWSVR, NLTWSVR and GLTWSVR methods for uniform and Gaussian additive noises over test set were obtained and illustrated in Figures 5.2(a) and 5.2(b) respectively.

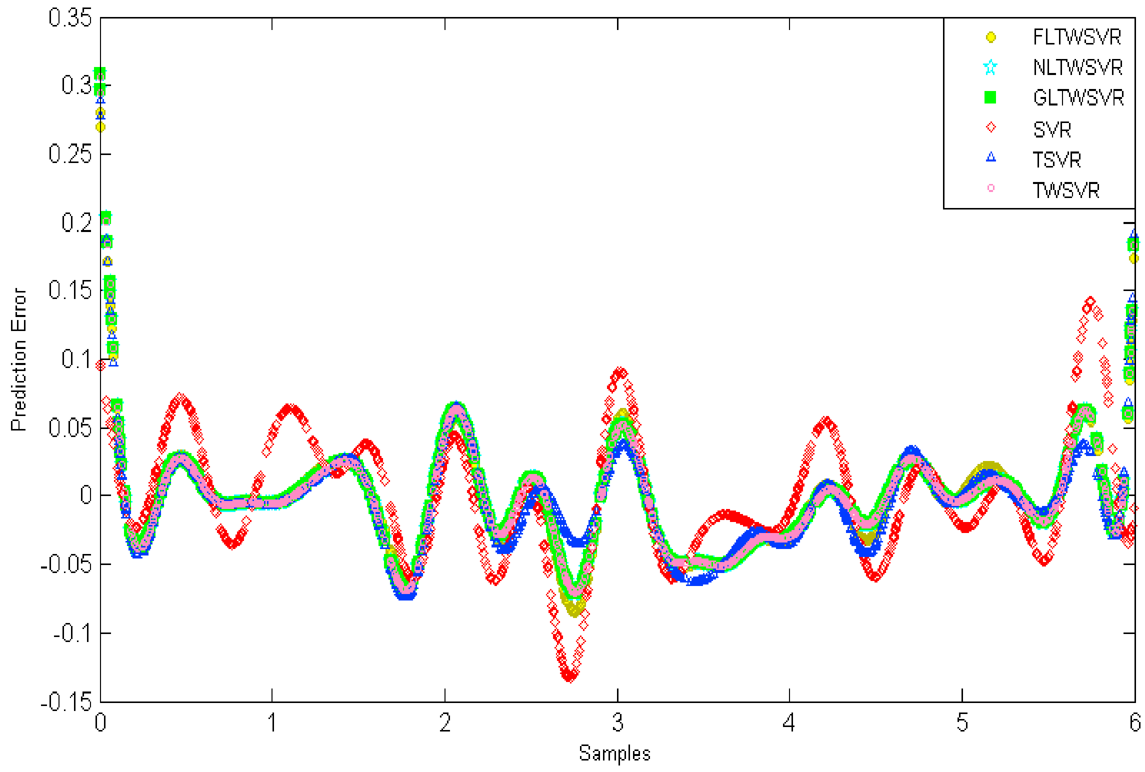


a) Uniform noise form [-0.2,0.2]

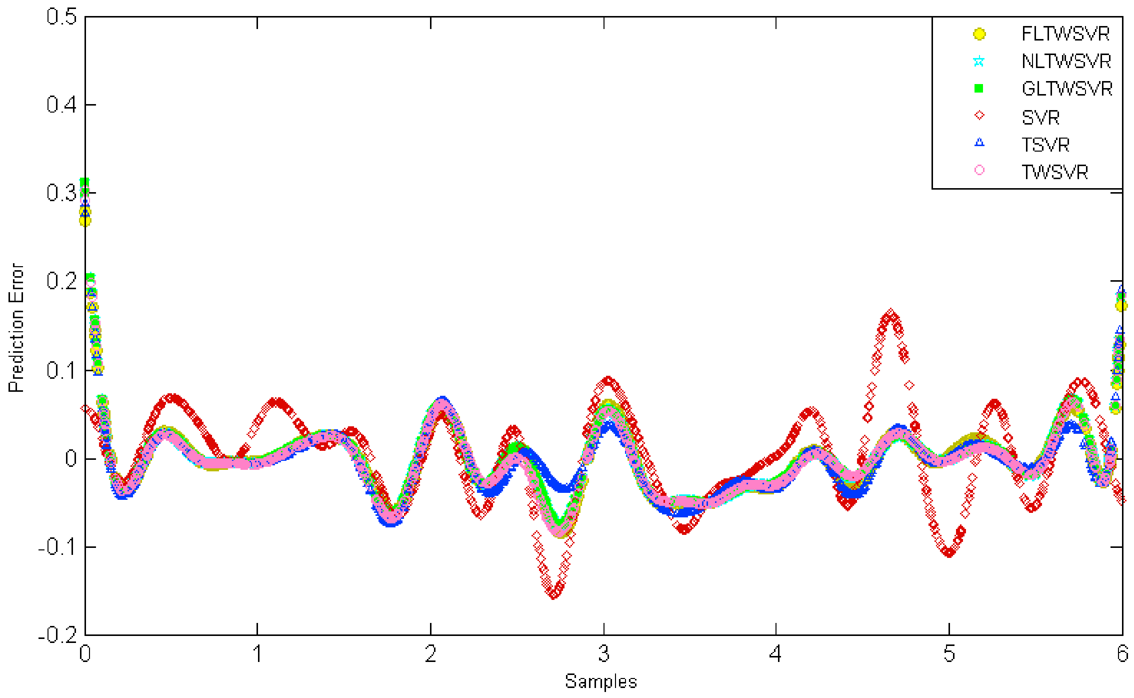


b) Gaussian noise with mean Zero and standard deviation 0.2

Figure 5.1: Results of approximation of $\sin(x)\cos(x^2)$ by SVR, TSVR, TWSVR and proposed methods: FLTWSVR, NLTWSVR, GLTWSVR on testing set. Gaussian kernel was employed.



a) Uniform noise from [-0.2,0.2]



b) Gaussian noise with mean zero and standard deviation 0.2

Figure 5.2: Prediction Error over the test set by SVR, TSVR, TWSVR, FLTWSVR, NLTWSVR and GLTWSVR for the dataset generated by the function $\sin(x)\cos(x^2)$. Gaussian kernel was employed.

Table 5.2: Performance comparison of our proposed methods: FLTWSVR, NLTWSVR and GLTWSVR with SVR, TSVR and TWSVR on synthetic datasets for uniform and Gaussian additive noises. RMSE was used for comparison. Gaussian kernel was employed. Time is for training in seconds. Bold type shows the best result.

a) Uniform noises from [-0.2, 0.2]

Dataset (Train Size, Test Size)	**SVR** (C,μ,ε) Time	**TSVR** $(C_1=C_2, \mu, \varepsilon_1=\varepsilon_2)$ Time	**TWSVR** $(C_1=C_2, \mu, \varepsilon_1=\varepsilon_2)$ Time	**FLTWSVR** $(C_1=C_2, \mu, \varepsilon_1=\varepsilon_2)$ Time	**NLTWSVR** $(C_1=C_2, \mu, \varepsilon_1=\varepsilon_2)$ Time	**GLTWSVR** $(C_1=C_2, \mu, \varepsilon_1=\varepsilon_2)$ Time
Function 1 (200 X 1,1000 X 1)	0.0444 $(10^1,2^3,10^{-1})$ 0.1755	0.0352 $(10^2,2^3,10^{-3})$ 0.1355	0.0358 $(10^1,2^3,10^{-1})$ 0.1270	0.0377 $(10^2,2^3,10^{-2})$ 0.0212	**0.0350** $(10^3,2^3,10^{-2})$ 0.0717	**0.0350** $(10^3,2^3,10^{-2})$ 0.0719
Function 2 (200 X2,1000 X 2)	0.0715 $(10^1,2^1,10^{-1})$ 0.1148	0.0643 $(10^5,2^1,10^{-1})$ 0.0875	0.0644 $(10^5,2^1,10^{-2})$ 0.0930	**0.0584** $(10^2,2^1,10^{-2})$ 0.0128	0.0592 $(10^2,2^1,10^{-3})$ 0.0443	0.0592 $(10^2,2^1,10^{-3})$ 0.0407
Function 3 (200 X 1,1000 X 1)	**0.0568** $(10^5,2^0,10^{-1})$ 0.2286	0.0599 $(10^0,2^2,10^{-1})$ 0.0632	0.0598 $(10^1,2^2,10^{-1})$ 0.0662	0.0595 $(10^4,2^1,10^{-2})$ 0.0185	0.0597 $(10^2,2^2,10^{-1})$ 0.0391	0.0597 $(10^2,2^2,10^{-1})$ 0.0383
Function 4 (200 X 2,1000 X 2)	**0.0778** $(10^0,2^{-2},10^{-1})$ 0.1084	0.0902 $(10^{-5},2^{-3},10^{-3})$ 0.0964	0.0889 $(10^{-1},2^{-3},10^{-1})$ 0.1351	0.0901 $(10^0,2^{-3},10^{-1})$ 0.0122	0.0901 $(10^0,2^{-3},10^{-1})$ 0.0226	0.0901 $(10^0,2^{-3},10^{-1})$ 0.0236
Function 5 (200 X2,1000 X 2)	**0.3609** $(10^2,2^5,10^{-2})$ 0.1235	0.4082 $(10^5,2^5,10^{-1})$ 0.1087	0.4083 $(10^1,2^5,10^{-3})$ 0.0734	0.3661 $(10^5,2^5,10^{-2})$ 0.0131	0.4110 $(10^1,2^5,10^{-3})$ 0.0330	0.4110 $(10^1,2^5,10^{-3})$ 0.0324

b) Gaussian noise with mean zero and standard deviation 0.2

Dataset (Train Size, Test Size)	**SVR** (C,μ,ε) Time	**TSVR** $(C_1=C_2, \mu, \varepsilon_1=\varepsilon_2)$ Time	**TWSVR** $(C_1=C_2, \mu, \varepsilon_1=\varepsilon_2)$ Time	**FLTWSVR** $(C_1=C_2, \mu, \varepsilon_1=\varepsilon_2)$ Time	**NLTWSVR** $(C_1=C_2, \mu, \varepsilon_1=\varepsilon_2)$ Time	**GLTWSVR** $(C_1=C_2, \mu, \varepsilon_1=\varepsilon_2)$ Time
Function 1 (200 X 1,1000 X 1)	0.0987 $(10^0,2^4,10^{-1})$ 0.0690	0.0853 $(10^5,2^3,10^{-1})$ 0.3106	**0.0678** $(10^0,2^3,10^{-1})$ 0.0630	0.0783 $(10^2,2^3,10^{-1})$ 0.0150	0.0801 $(10^2,2^3,10^{-1})$ 0.3270	0.0801 $(10^2,2^3,10^{-1})$ 0.4522
Function 2 (200 X 2,1000 X 2)	**0.1003** $(10^5,2^{-4},10^{-2})$ 0.1153	0.1084 $(10^0,2^{-1},10^{-1})$ 0.0581	0.1085 $(10^0,2^{-1},10^{-3})$ 0.0565	0.1048 $(10^2,2^{-1},10^{-3})$ 0.0125	0.1046 $(10^1,2^{-1},10^{-2})$ 0.0413	0.1046 $(10^1,2^{-1},10^{-2})$ 0.0400
Function 3 (200 X 1,1000 X 1)	0.0992 $(10^4,2^0,10^{-3})$ 0.1855	0.0736 $(10^{-1},2^2,10^{-3})$ 0.0574	0.0737 $(10^{-1},2^2,10^{-2})$ 0.0604	**0.0725** $(10^{-5},2^2,10^{-2})$ 0.0166	**0.0725** $(10^{-5},2^2,10^{-2})$ 0.0275	**0.0725** $(10^{-5},2^2,10^{-2})$ 0.0269
Function 4 (200 X 2,1000 X 2)	0.1444 $(10^1,2^{-4},10^{-2})$ 0.1742	0.1339 $(10^{-1},2^{-4},10^{-1})$ 0.0615	0.1339 $(10^{-1},2^{-4},10^{-3})$ 0.0565	**0.1334** $(10^{-3},2^{-4},10^{-3})$ 0.0166	**0.1334** $(10^{-3},2^{-4},10^{-3})$ 0.0269	**0.1334** $(10^{-3},2^{-4},10^{-3})$ 0.0269
Function 5 (200 X 2,1000 X 2)	0.4989 $(10^2,2^4,10^{-3})$ 0.1173	0.5782 $(10^{-1},2^5,10^{-3})$ 0.0584	0.5782 $(10^{-1},2^5,10^{-3})$ 0.0563	**0.3584** $(10^5,2^4,10^{-2})$ 0.0177	0.5769 $(10^0,2^5,10^{-3})$ 0.0410	0.5769 $(10^0,2^5,10^{-3})$ 0.0413

5.3 Real-world Benchmark Datasets

In this section, to further test the effectiveness of LTWSVR algorithms compared to SVR, TSVR and TWSVR, we illustrate the experiments performed both linearly and nonlinearly on several well known real-world datasets. For this we use 27 real-world datasets: Hydraulic actuator (Gretton et al., 2001; Sjoberg et al., 1995); Gas Furnace (Box and Jenkins, 1976); Pyrim, Servo, Triazines, Wisconsin breast cancer, Boston, Forest fires, Concrete CS, Wine quality red, Concrete Slump and AutoPrice datasets form UCI repository (Murphy and Aha, 1992); Flexible robotic arm (http://homes.esat.kuleuven.be/~smc/daisy/daisy data.html); Pollution, NO2, Bodyfat, Balloon and Quake (http://lib.stat.cmu.edu/ datasets); Motorcycle (Eubank, 1999); Demo (DELVE, 2005); Sunspots times series dataset (http://www.bme.ogi.edu/~ericwan/data.html); IBM, Standard & Poor 500 (SNP500), Citigroup, Intel, Microsoft and RedHat financial time series datasets (http://finance.yahoo.com).

For all experiments, first all samples are normalized before learning as follows:

$$\hat{x}_{ij} = \frac{x_{ij} - x_j^{\min}}{x_j^{\max} - x_j^{\min}}$$

where x_{ij} is $(i,j)^{th}$ entry in the input matrix $A_{m\times n}$, $\hat{x}_{ij}$ its corresponding estimated normalized value and $x_j^{\min} = \min\limits_{i\in m}(x_{ij})$ and $x_j^{\min} = \min\limits_{j\in n}(x_{ij}),\ j = 1,\ldots,n$ denote minimum and maximum values in the j^{th} column respectively. Second, optimal parameters are determined by performing ten-fold cross-validation on training set as whole dataset. As for testing, we apply the cross-validation by taking random ninety percent of the dataset for training and remaining for testing. Repeating this process ten times and taking their average, test accuracy is determined.

The Hydraulic actuator dataset is taken as the first example for our experiment. It has been widely used in nonlinear system identification (Gretton et al., 2001; Sjoberg et al., 1995). It contains 1024 samples with input variable $u(t)$ and the output variable $y(t)$ denotes the valve position and oil pressure respectively. For the purpose of comparison, 1021 samples with five attributes are taken of the form $(\mathbf{x}(t), y(t))$ where

$\mathbf{x}(t) = [y(t-1), y(t-2), y(t-3), u(t-1), u(t-2)]^t$.

As for second example, frequently used for nonlinear identification problems, Box and Jenkins gas furnace dataset is taken. It is a time series dataset which contains 296 samples with input variable $u(t)$ and output variable $y(t)$ denote gas flow rate and CO_2 concentration respectively. In experiment 293 samples with six attributes of the form: $(\mathbf{x}(t), y(t))$ where $\mathbf{x}(t) = [y(t-1), y(t-2), y(t-3), u(t-1), u(t-2), u(t-3)]^t$, are taken for testing. The prediction accuracy and prediction error plots over whole dataset employing linear kernel by SVR, TSVR, TWSVR, FLTWSVR, NLTWSVR and GLTWSVR are shown in Figures 5.3 and 5.4 respectively. Using Gaussian kernel, prediction accuracy and prediction error over the whole dataset by SVR, TSVR, TWSVR, FLTWSVR, NLTWSVR and GLTWSVR are shown in Figures 5.5 and 5.6 respectively.

As an interesting example, the flexible robotic arm, in estimation of the inverse dynamics of a flexible robot arm (Souza et. al. 2006), is taken. The dynamics of robot arm is modeled as a transfer function of the measured values of the reaction torque of the structure (input time series, $u(t)$) whose output $y(t)$ is its corresponding acceleration. Following the work of (Souza and Barreto, 2006), samples are taken to be of the form: $(\mathbf{x}(t), x^{out}(t))$ where $\mathbf{x}(t) = [u(t-1), \ldots, u(t-5), y(t-1), \ldots y(t-4)]^t$ and $x^{out}(t) = u(t)$.

In addition to the above datasets, experiments are performed on other well known datasets: Bodyfat, NO2, Balloon, Pollution and Quake available from Statlib collection http://lib.stat.cmu.edu/datasets. Bodyfat is a real dataset lists estimates of the percentage of body fat of 252 peoples having from body density values. NO2 dataset contains 500 sample from a dataset with seven variables collected by the Norwegian Public Roads Administration (Vlachos, 2005). Balloon dataset contains 2001 observations of radiation having trend and outliers. The pollution dataset lists an estimate relating air pollution to mortality. Quake dataset contains 2178 samples with three attributes (focal depth, latitude and longitude), lists information for earthquakes occurred between January 1964 and February 1986.

Another popular benchmark dataset, Motorcycle consists of a series of accelerometer readings over time in a simulation of motorcycle accidents used to test crash-helmets (Silverman, 1985). The Demo dataset (DELVE, 2005) which consists of 294 samples artificially generated from a distribution based on assumptions and notions concerning the relationships between people's sex, age, number of siblings, income, and favorite colour. The Sunspots (http://www.bme.ogi. edu/~ericwan/data.html) time series dataset, containing 295 yearly readings (year 1700 to 1994) but only 290 samples taken as a whole because current value is predicted from five previous values.

To further test the performance of algorithms, we evaluated them on several publicly available datasets from UCI repository including Pyrim, Servo, Triazines, Wisconsin breast cancer, Boston, Forest fires, Concrete CS, Wine quality red, Concrete Slump and AutoPrice datasets. These datasets are commonly used in testing regression algorithms.

Finally, as examples of financial time series datasets, the stock index of Citigroup, Intel, Microsoft and RedHat are considered. These datasets contain information about 755 closing stock prices (01-01-2006 to 31-12-2008). Since the current value is predicted from five previous values so only 750 samples taken as a whole dataset.

5.3.1 Numerical experiment using linear regressors

In this sub-section, all the experiments are performed using the linear kernel. In order to evaluate the performance of the LTWSVR algorithms (FLTWSVR,NLTWSVR, GNLTWSVR) with SVR, TSVR and TWSVR, we obtained optimal parameter values by performing ten- fold cross validation for each dataset and computed learning time, average RMSE and standard deviation summarized in Table 5.3. As seen from Table 5.3, for most of the cases, LTWSVR algorithms derive better generalization performance than SVR, TSVR and TWSVR. As for training time, LTWSVR algorithms spend the least CPU time among all the methods.

To analyze the performance of all the six algorithms over multiple datasets, we used Friedman test with post hoc test which is stated as a simple, safe and robust non-parametric test (Demsar, 2006). For this, we computed average ranks of these algorithms on RMSE values which are listed in Table 5.4. Under the null hypothesis that all the algorithms are identical, Friedman statistics can be computed as follows:

$$\chi_F^2 = \frac{12\times 27}{6\times 7}\left[(4.7777^2 + 4.0925^2 + 4.037^2 + 2.6296^2 + 2.7777^2 + 2.6851^2) - \frac{6\times 7^2}{4}\right]$$
$$\cong 32.4965$$

$$F_F = \frac{26\times 32.4965}{27\times 5 - 32.4965} \cong 8.2427.$$

where F_F is distributed according to $F-$distribution with $(5, 130)$ degrees of freedom. The critical value of $F(5, 130)$ is 2.2839 for the level of significance $\alpha = 0.05$ and similarly 1.8920 for $\alpha = 0.10$. Since F_F is greater than both critical values, so we reject the null hypothesis. We use the Nemenyi test for further pair wise comparison. According to (Demsar, 2006), the performance of two algorithms is significantly different if the corresponding average ranks differ by at least the critical difference, at $p = 0.10$ critical difference (CD) is $2.589\sqrt{\frac{6\times 7}{6\times 27}} \cong 1.3182.$ we have the following comparison results:

(i) For Absolute, Newton and Generalized; the difference of one algorithm with other two algorithms is less than the critical difference value. This indicates that the post hoc test fails to detect any significant difference among these three algorithms.

(ii) The Absolute method significantly performs better than the SVR $(4.7777 - 2.6296 = 2.1481 > 1.3182)$, the TSVR $(4.0925 - 2.6296 = 1.4629 > 1.3182)$ and the TWSVR $(4.0370 - 2.6296 = 1.4074 > 1.3182)$.

(iii) The Newton method significantly performs better than the SVR $(4.7777 - 2.7777 = 2 > 1.3182)$. There is no any significant difference detected for the Newton method compared with the TSVR $(4.0925 - 2.7777 = 1.3148 < 1.3182)$ and the TWSVR $(4.0370 - 2.7777 = 1.2593 < 1.3182)$.

(iv) The generalized method significantly performs better than the SVR $(4.7777 - 2.6851 = 2.0926 > 1.3182)$, the TSVR $(4.0925 - 2.6851 = 1.4074 > 1.3182)$ and the TWSVR $(4.0370 - 2.6851 = 1.3519 > 1.3182)$.

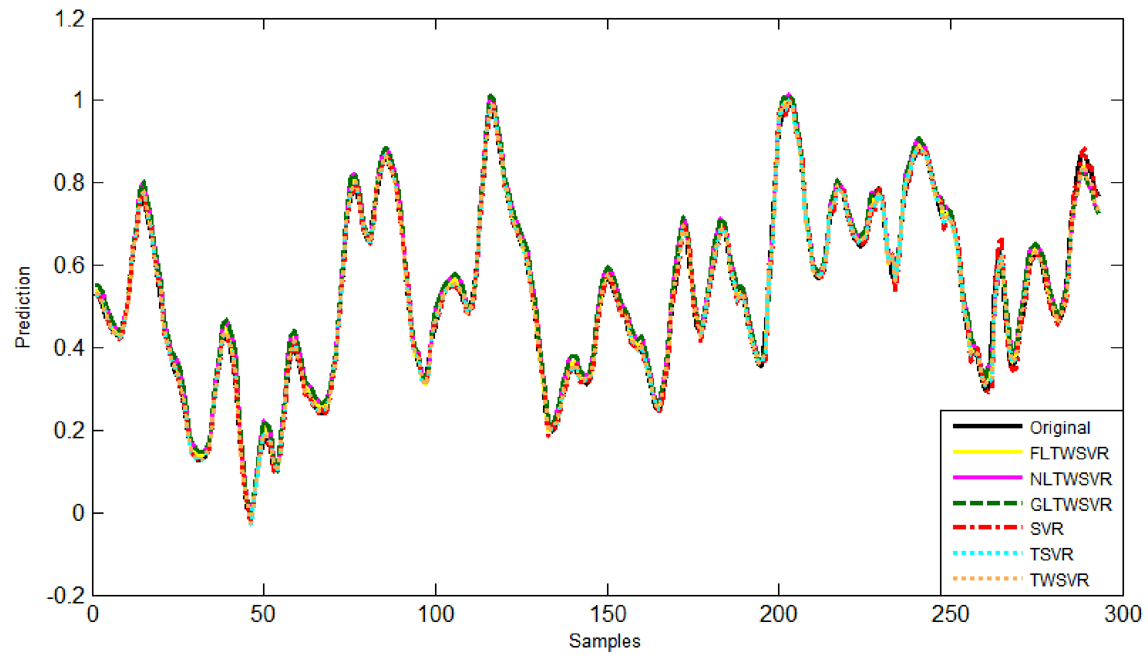


Figure 5.3: Result of comparison on Gas furnace dataset. Linear kernel was employed

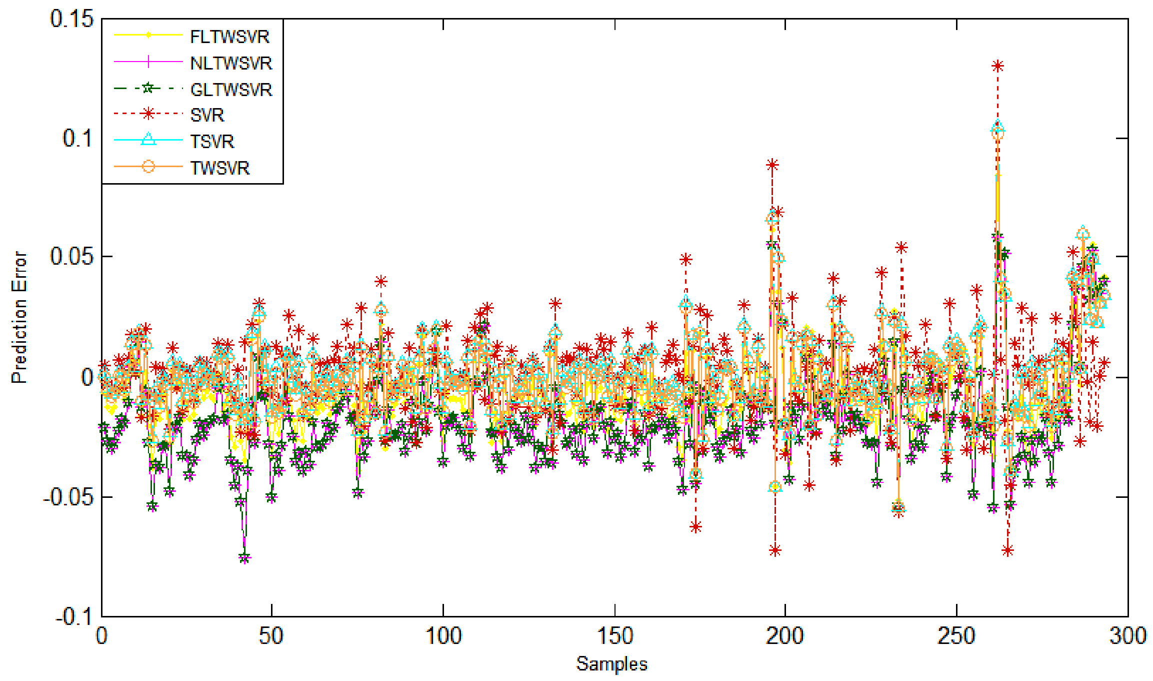


Figure 5.4: Prediction Error over the whole dataset by SVR, TSVR, TWSVR, FLTWSVR, NLTWSVR and GLTWSVR for the gas furnace dataset of Box-Jenkins. Linear kernel was employed.

Table 5.3: Performance comparison of our proposed methods: FLTWSVR, NLTWSVR and GLTWSVR with SVR, TSVR and TWSVR. RMSE was used for comparison. Linear kernel was employed. Bold type shows the best result.

Dataset (Train Size, Test Size)	**SVR** (C, ε) Time	**TSVR** $(C_1=C_2, \varepsilon_1=\varepsilon_2)$ Time	**TWSVR** $(C_1=C_2, \varepsilon_1=\varepsilon_2)$ Time	**FLTWSVR** $(C_1=C_2, \varepsilon_1=\varepsilon_2)$ Time	**NLTWSVR** $(C_1=C_2, \varepsilon_1=\varepsilon_2)$ Time	**GLTWSVR** $(C_1=C_2, \varepsilon_1=\varepsilon_2)$ Time
Hydraulic actuator (1021 X 5)	0.0145±0.0046 $(10^{1},10^{-2})$ 4.3804	0.0134±0.0034 $(10^{-2},10^{-3})$ 1.6917	0.0135±0.0034 $(10^{-2},10^{-3})$ 1.7299	0.0133±0.0038 $(10^{-5},10^{-3})$ 0.1617	**0.0132±0.0041** $(10^{-5},10^{-3})$ 0.3350	0.0133±0.0038 $(10^{-4},10^{-3})$ 0.6250
Gas furnace (293 X 6)	0.0199±0.0068 $(10^{2},10^{-2})$ 0.1438	0.0168±0.0040 $(10^{-1},10^{-3})$ 0.0989	0.0167±0.0041 $(10^{-1},10^{-3})$ 0.0980	**0.0165±0.0045** $(10^{5},10^{-1})$ 0.0099	**0.0165±0.0045** $(10^{5},10^{-1})$ 0.0111	0.0166±0.0047 $(10^{5},10^{-1})$ 0.0116
Pyrim (74 X 26)	**0.1056±0.0577** $(10^{0},10^{-2})$ 0.0080	0.1268±0.0725 $(10^{1},10^{-3})$ 0.0120	0.1257±0.0717 $(10^{3},10^{-2})$ 0.0160	0.1183±0.0640 $(10^{5},10^{-1})$ 0.0008	0.1241±0.0727 $(10^{4},10^{-2})$ 0.0054	0.1241±0.0727 $(10^{4},10^{-2})$ 0.0878
Servo (167 X 4)	0.2288±0.1025 $(10^{4},10^{-1})$ 0.0411	0.1608±0.0372 $(10^{-2},10^{-3})$ 0.0289	**0.1605±0.0372** $(10^{-1},10^{-1})$ 0.0279	0.1608±0.0361 $(10^{0},10^{-3})$ 0.0021	0.1608±0.0361 $(10^{0},10^{-3})$ 0.0082	0.1608±0.0361 $(10^{0},10^{-3})$ 0.0083
Triazines (186 X 58)	0.2217±0.0424 $(10^{0},10^{-1})$ 0.0512	**0.2063±0.0726** $(10^{-1},10^{-3})$ 0.0387	0.2104±0.0754 $(10^{0},10^{-1})$ 0.0356	0.2073±0.0703 $(10^{0},10^{-2})$ 0.0033	0.2073±0.0703 $(10^{0},10^{-2})$ 0.0114	0.2073±0.0703 $(10^{0},10^{-2})$ 0.0191
Wisconsin B.C. (194 X 34)	**0.1835±0.0558** $(10^{-1},10^{-1})$ 0.0498	0.1887±0.0405 $(10^{0},10^{-3})$ 0.0391	0.1887±0.0404 $(10^{0},10^{-3})$ 0.0395	0.1894±0.0613 $(10^{-5},10^{-3})$ 0.0038	0.1894±0.0613 $(10^{-5},10^{-3})$ 0.0077	0.1894±0.0613 $(10^{-4},10^{-3})$ 0.0083
Boston (506 X 13)	0.1130±0.0434 $(10^{0},10^{-2})$ 0.5735	0.1076±0.0207 $(10^{-1},10^{-3})$ 0.2853	0.1076±0.0206 $(10^{-1},10^{-3})$ 0.2876	**0.1061±0.0265** $(10^{-5},10^{-3})$ 0.0269	0.1062±0.0261 $(10^{-5},10^{-3})$ 0.1032	**0.1061±0.0265** $(10^{-4},10^{-3})$ 0.1027
Forest fires (517 X 12)	0.0441±0.0413 $(10^{-5},10^{-3})$ 0.6717	**0.0415±0.0428** $(10^{-1},10^{-1})$ 0.3425	0.0416±0.0427 $(10^{-1},10^{-3})$ 0.3476	0.0436±0.0409 $(10^{-5},10^{-3})$ 0.0276	0.0436±0.0409 $(10^{-5},10^{-3})$ 0.0790	0.0437±0.0408 $(10^{-4},10^{-3})$ 0.1075
ConcreteCS (1030 X 8)	0.1305±0.0094 $(10^{1},10^{-1})$ 3.8649	0.1328±0.0076 $(10^{1},10^{-3})$ 2.6391	0.1306±0.0082 $(10^{0},10^{-1})$ 2.0905	**0.1304±0.0104** $(10^{1},10^{-3})$ 0.2377	0.1305±0.0055 $(10^{1},10^{-3})$ 1.6756	0.1305±0.0055 $(10^{1},10^{-3})$ 1.6685
Wine quality red (1599 X 11)	0.1423±0.0118 $(10^{-1},10^{-1})$ 12.389	0.1304±0.0059 $(10^{-1},10^{-1})$ 5.0902	0.1303±0.0059 $(10^{-1}$,10-3) 5.1135	0.1302±0.0070 $(10^{-5},10^{-3})$ 0.5791	0.1301±0.0064 $(10^{-5},10^{-3})$ 2.2136	**0.1300±0.0096** $(10^{-4},10^{-3})$ 2.2322
Concrete Slump (103 X 10)	0.0612±0.0161 $(10^{0},10^{-2})$ 0.0131	0.0595±0.0150 $(10^{-5},10^{-1})$ 0.0145	0.0600±0.0152 $(10^{-1},10^{-2})$ 0.0148	**0.0594±0.0151** $(10^{0},10^{-2})$ 0.0008	0.0596±0.0180 $(10^{0},10^{-2})$ 0.0029	**0.0594±0.0151** $(10^{0},10^{-2})$ 0.0029
Auto price (159 X 15)	0.0879±0.0259 $(10^{-1},10^{-3})$ 0.0306	0.0898±0.0253 $(10^{3},10^{-1})$ 0.0379	0.0898±0.0253 $(10^{1},10^{-3})$ 0.0302	**0.0856±0.0225** $(10^{3},10^{-3})$ 0.0024	0.0865±0.0226 $(10^{3},10^{-2})$ 0.0299	0.0861±0.0238 $(10^{2},10^{-2})$ 0.0150
Flexible robotic arm (1019 X 9)	0.0150±0.0007 $(10^{2},10^{-2})$ 4.2171	0.0149±0.0005 $(10^{-1},10^{-1})$ 1.8890	**0.0148±0.0006** $(10^{-1},10^{-2})$ 1.9419	**0.0148±0.0005** $(10^{1},10^{-3})$ 0.2334	**0.0148±0.0005** $(10^{1},10^{-3})$ 1.6090	**0.0148±0.0005** $(10^{1},10^{-3})$ 1.5877

Pollution (60 X 15)	**0.1175±0.0498** (10^{-1},10^{-2}) 0.0059	0.1263±0.0495 (10^{3},10^{-3}) 0.0269	0.1264±0.0495 (10^{1},10^{-3}) 0.0098	0.1209±0.0497 (10^{5},10^{-2}) 0.0005	0.1261±0.0494 (10^{3},10^{-3}) 0.0035	0.1245±0.0489 (10^{2},10^{-3}) 0.0025
NO2 (500 X 7)	0.1025±0.0189 (10^{5},10^{-1}) 0.5956	0.1020±0.0097 (10^{-1},10^{-1}) 0.2683	0.1020±0.0097 (10^{-1},10^{-3}) 0.2631	0.1016±0.0126 (10^{-5},10^{-3}) 0.0261	**0.1010±0.0170** (10^{-5},10^{-3}) 0.0985	**0.1010±0.0170** (10^{-4},10^{-3}) 0.0951
Bodyfat (252 X 14)	0.0504±0.0486 (10^{2},10^{-2}) 0.1075	**0.0234±0.0257** (10^{-1},10^{-1}) 0.0737	0.0235±0.0256 (10^{-1},10^{-3}) 0.0725	0.0257±0.0245 (10^{-5},10^{-3}) 0.0041	0.0257±0.0245 (10^{-5},10^{-3}) 0.0069	0.0257±0.0245 (10^{-4},10^{-3}) 0.0132
Balloon (2001 X 1)	0.0552±0.0046 (10^{-2},10^{-1}) 20.444	0.0512±0.0061 (10^{0},10^{-3}) 17.161	0.0512±0.0061 (10^{0},10^{-3}) 17.090	**0.0511±0.0057** (10^{1},10^{-2}) 1.2527	**0.0511±0.0057** (10^{1},10^{-2}) 10.010	**0.0511±0.0057** (10^{1},10^{-2}) 11.236
Quake (2178 X 3)	0.1792±0.0103 (10^{4},10^{-1}) 39.092	**0.1718±0.0091** (10^{-5},10^{-1}) 12.358	**0.1718±0.0091** (10^{-3},10^{-2}) 11.007	**0.1718±0.0091** (10^{-1},10^{-3}) 1.3326	**0.1718±0.0091** (10^{-1},10^{-3}) 7.6448	**0.1718±0.0091** (10^{-2},10^{-3}) 5.2452
Motorcycle (133 X 1)	0.2909±0.0542 (10^{0},10^{-1}) 0.0163	0.2224±0.0269 (10^{-1},10^{-1}) 0.0185	0.2212±0.0278 (10^{-1},10^{-3}) 0.0187	0.2211±0.0269 (10^{-5},10^{-3}) 0.0011	**0.2202±0.0299** (10^{-5},10^{-3}) 0.0033	**0.2202±0.0299** (10^{-4},10^{-3}) 0.0033
Demo (2048 X 4)	0.1026±0.0110 (10^{0},10^{-1}) 33.092	**0.0997±0.0118** (10^{-3},10^{-3}) 9.6141	**0.0997±0.0118** (10^{-3},10^{-3}) 9.7527	**0.0997±0.0118** (10^{-5},10^{-3}) 1.1899	**0.0997±0.0118** (10^{-5},10^{-3}) 4.4432	**0.0997±0.0118** (10^{-4},10^{-3}) 5.3983
Sunspots (290 X 5)	0.0940±0.0219 (10^{2},10^{-1}) 0.1323	0.0882±0.0154 (10^{-1},10^{-1}) 0.0852	0.0881±0.0199 (10^{-1},10^{-3}) 0.0829	0.0881±0.0195 (10^{-5},10^{-3}) 0.0075	**0.0879±0.0174** (10^{-5},10^{-3}) 0.0204	**0.0879±0.0174** (10^{-4},10^{-3}) 0.0255
IBM (750 X 5)	0.0272±0.0032 (10^{5},10^{-2}) 1.9445	0.0270±0.0021 (10^{-2},10^{-3}) 0.6950	0.0270±0.0030 (10^{-1},10^{-3}) 0.8688	**0.0269±0.0035** (10^{-5},10^{-3}) 0.0736	**0.0269±0.0035** (10^{-5},10^{-3}) 0.1471	**0.0269±0.0035** (10^{-4},10^{-3}) 0.2760
SNP500 (750 X 5)	**0.0222±0.0029** (10^{0},10^{-3}) 1.5770	0.0223±0.0029 (10^{-1},10^{-3}) 0.8678	0.0223±0.0029 (10^{-1},10^{-3}) 0.8637	**0.0222±0.0033** (10^{-5},10^{-1}) 0.0744	**0.0222±0.0033** (10^{-5},10^{-1}) 0.1140	**0.0222±0.0033** (10^{-4},10^{-1}) 0.1126
Citigroup (750 X 5)	**0.0149±0.0013** (10^{4},10^{-2}) 1.6472	**0.0149±0.0013** (10^{-1},10^{-3}) 0.9193	**0.0149±0.0013** (10^{-1},10^{-3}) 0.9113	**0.0149±0.0013** (10^{-3},10^{-1}) 0.0713	**0.0149±0.0013** (10^{-2},10^{-1}) 0.0991	**0.0149±0.0013** (10^{1},10^{-1}) 0.0950
Intel (750 X 5)	0.0294±0.0042 (10^{2},10^{-2}) 1.5290	**0.0293±0.0049** (10^{-1},10^{-1}) 0.8357	**0.0293±0.0049** (10^{-1},10^{-3}) 0.8333	**0.0293±0.0049** (10^{-5},10^{-3}) 0.0715	**0.0293±0.0049** (10^{-5},10^{-3}) 0.1467	**0.0293±0.0049** (10^{-4},10^{-3}) 0.2744
Microsoft (750 X 5)	**0.0279±0.0050** (10^{2},10^{-3}) 1.5535	0.0281±0.0029 (10^{-1},10^{-3}) 0.8386	0.0281±0.0029 (10^{-1},10^{-3}) 0.8441	**0.0279±0.0047** (10^{-5},10^{-3}) 0.0750	**0.0279±0.0047** (10^{-5},10^{-3}) 0.1503	**0.0279±0.0047** (10^{-4},10^{-3}) 0.2779
RedHat (750 X 5)	**0.0254±0.0052** (10^{0},10^{-3}) 1.5655	0.0255±0.0047 (10^{-1},10^{-1}) 0.8445	0.0255±0.0047 (10^{-1},10^{-3}) 0.8315	**0.0254±0.0050** (10^{-5},10^{-3}) 0.0716	**0.0254±0.0050** (10^{-5},10^{-3}) 0.1503	**0.0254±0.0050** (10-4,10^{-3}) 0.2800

Table 5.4: Average ranks of SVR, TSVR, TWSVR, FLTWSVR, NLTWSVR and GLTWSVR with linear kernel.

Dataset	**SVR**	**TSVR**	**TWSVR**	**FLTWSVR**	**NLTWSVR**	**GLTWSVR**
Hydraulic actuator	6	4	5	2.5	1	2.5
Gas furnace	6	5	4	1.5	1.5	3
Pyrim	1	6	5	2	3.5	3.5
Servo	6	3.5	1	3.5	3.5	3.5
Triazines	6	1	5	3	3	3
Wisconsin B.C.	1	2.5	2.5	5	5	5
Boston	6	4.5	4.5	1.5	3	1.5
Forest fires	6	1	2	3.5	3.5	5
ConcreteCS	3	6	5	1	3	3
Wine quality red	6	5	4	3	2	1
Concrete Slump	6	3	5	1.5	4	1.5
Auto price	4	5.5	5.5	1	3	2
Flexible robotic arm	6	5	2.5	2.5	2.5	2.5
Pollution	1	5	6	2	4	3
NO2	6	4.5	4.5	3	1.5	1.5
Bodyfat	6	1	2	4	4	4
Balloon	6	4.5	4.5	2	2	2
Quake	6	3	3	3	3	3
Motorcycle	6	5	4	3	1.5	1.5
Demo	6	3	3	3	3	3
Sunspots	6	5	3.5	3.5	1.5	1.5
IBM	6	4.5	4.5	2	2	2
Snp500	2.5	5.5	5.5	2.5	2.5	2.5
Citigroup	3.5	3.5	3.5	3.5	3.5	3.5
Intel	6	3	3	3	3	3
Microsoft	2.5	5.5	5.5	2.5	2.5	2.5
RedHat	2.5	5.5	5.5	2.5	2.5	2.5
Average Rank	4.7777	4.0925	4.0370	**2.6296**	2.7777	2.6851

5.3.2 Numerical experiment using nonlinear regressors

In this sub-section, all the experiments are performed using the Gaussian kernel. Again to evaluate the performance of the LTWSVR algorithms (FLTWSVR, NLTWSVR, GLTWSVR) in comparison to SVR, TSVR and TWSVR; learning time, average RMSE and standard deviation are computed for each dataset and summarized in Table 5.5 along with optimal parameter value. For nonlinear case, one can observe from Table 5.5 that LTWSVR algorithms also spend least CPU time in comparison with other methods and has better generalization performance for most of the cases.

To analyze the performance of all the six algorithms over multiple datasets, we used Friedman test with post hoc test (Demsar, 2006) as we have done for the linear case. For this, we computed average ranks on RMSE values and are listed in Table 5.6. Under the null hypothesis that all the algorithms are identical, Friedman statistics can be computed as follows:

$$\chi_F^2 = \frac{12\times 27}{6\times 7}\left[(3.6851^2 + 4.1666^2 + 3.9259^2 + 2.8512^2 + 3.1481^2 + 3.2222^2) - \frac{6\times 7^2}{4}\right]$$
$$\cong 9.8407$$

$$F_F = \frac{26\times 9.8407}{27\times 5 - 9.8407} \cong 2.0442$$

where F_F is distributed according to $F-$distribution with $(5, 130)$ degrees of freedom. The critical value of $F(5, 130)$ is 2.2839 for the level of significance $\alpha = 0.05.$ Since F_F is smaller than critical value $(2.0442 < 2.2839),$ so there is no significant error between the algorithms.

Finally, numerical experiments performed for both linear and nonlinear cases validate that LTWSVR algorithms outperform the other three methods.

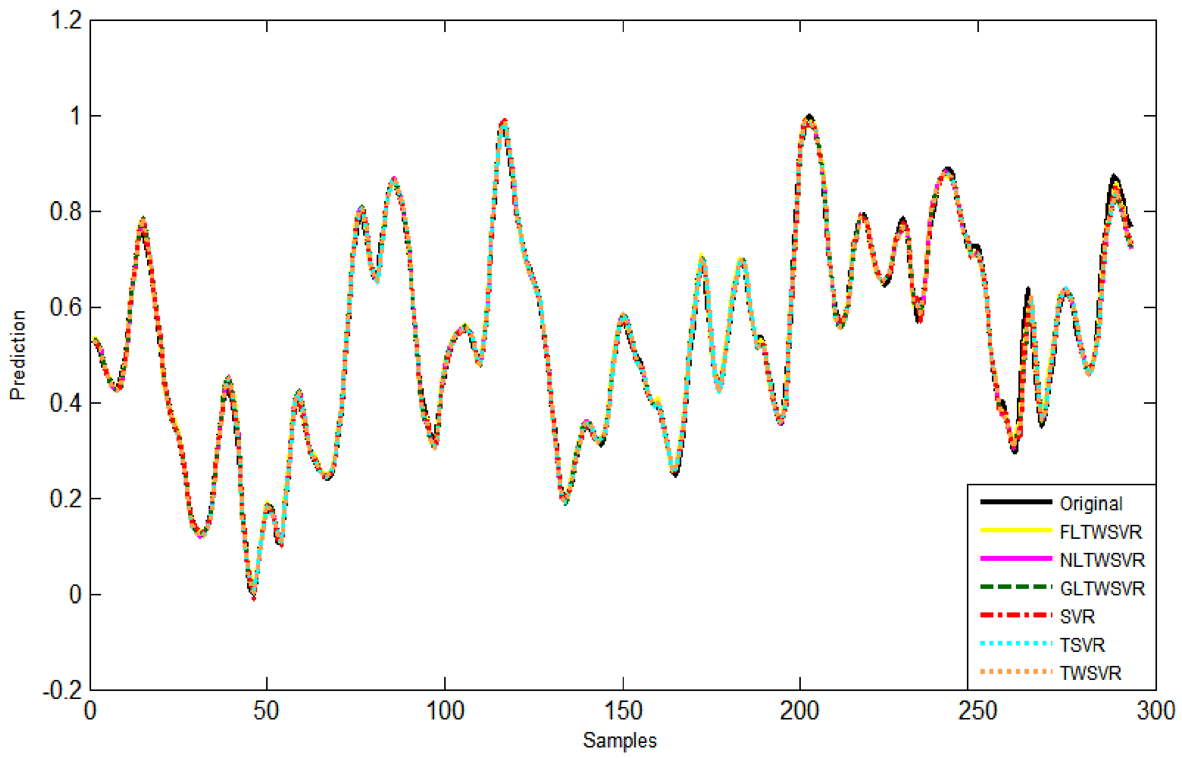


Figure 5.5: Result of comparison on Gas furnace dataset. Gaussian kernel was employed

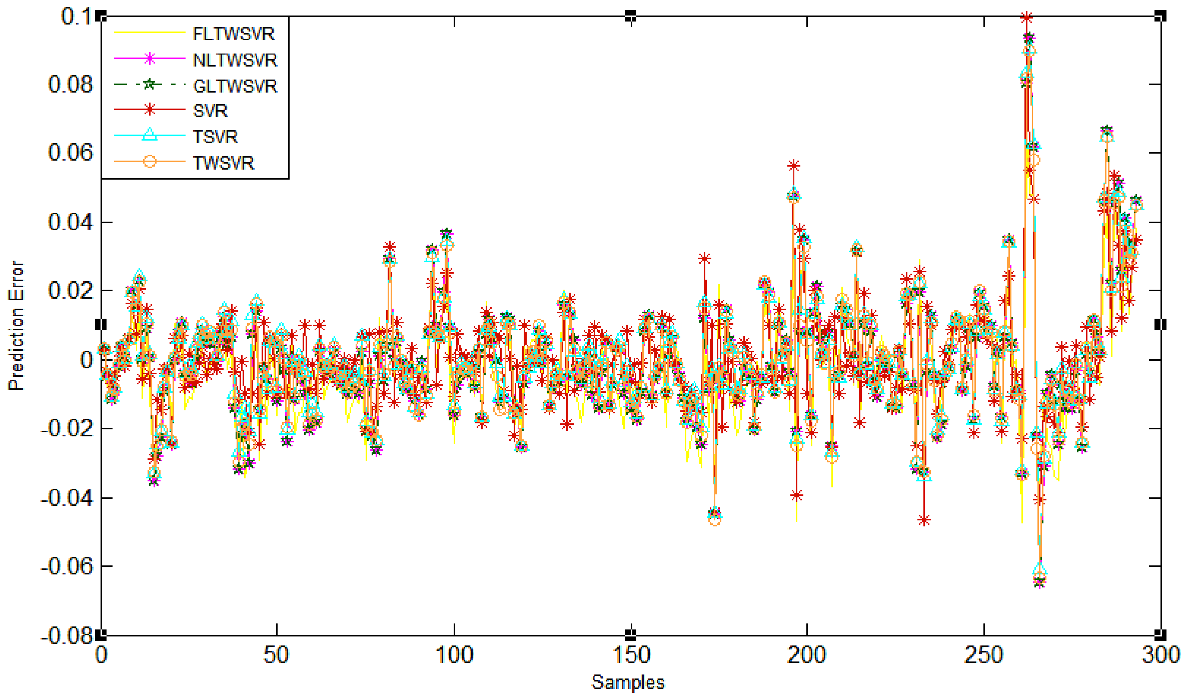


Figure 5.6: Prediction Error over the whole dataset by SVR, TSVR, TWSVR, FLTWSVR, NLTWSVR and GLTWSVR for the gas furnace dataset of Box-Jenkins. Gaussian kernel was employed.

Table 5.5: Performance comparison of our proposed methods: FLTWSVR, NLTWSVR and GLTWSVR with SVR, TSVR and TWSVR. RMSE was used for comparison. Gaussian kernel was employed. Bold type shows the best result.

Dataset (Train Size, Test Size)	**SVR** (C,μ,ε) Time	**TSVR** $(C_1=C_2, \mu, \varepsilon_1=\varepsilon_2)$ Time	**TWSVR** $(C_1=C_2, \mu, \varepsilon_1=\varepsilon_2)$ Time	**FLTWSVR** $(C_1=C_2, \mu, \varepsilon_1=\varepsilon_2)$ Time	**NLTWSVR** $(C_1=C_2, \mu, \varepsilon_1=\varepsilon_2)$ Time	**GLTWSVR** $(C_1=C_2, \mu, \varepsilon_1=\varepsilon_2)$ Time
Hydraulic actuator (1021 X 5)	0.0127±0.0041 $(10^{2},2^{1},10^{-3})$ 5.7962	0.0125±0.0026 $(10^{-2},2^{2},10^{-1})$ 2.3039	0.0126±0.0022 $(10^{-2},2^{2},10^{-3})$ 2.3074	0.0123±0.0034 $(10^{0},2^{2},10^{-3})$ 0.7374	**0.0121±0.0033** $(10^{0},2^{2},10^{-3})$ 1.7106	0.0123±0.0034 $(10^{0},2^{2},10^{-3})$ 1.7332
Gas furnace (293 X 6)	**0.0168±0.0042** $(10^{3},2^{-5},10^{-2})$ 0.1807	0.0192±0.0055 $(10^{-1},2^{-1},10^{-1})$ 0.1134	0.0192±0.0055 $(10^{-1},2^{-1},10^{-3})$ 0.1118	0.0184±0.0033 $(10^{5},2^{-2},10^{-3})$ 0.0284	0.0198±0.0058 $(10^{0},2^{-1},10^{-3})$ 0.0628	0.0195±0.0050 $(10^{0},2^{-1},10^{-3})$ 0.0654
Pyrim (74 X 26)	0.0788±0.0534 $(10^{0},2^{-3},10^{-2})$ 0.0092	0.0779±0.0579 $(10^{-5},2^{-2},10^{-3})$ 0.0116	0.0779±0.0579 $(10^{1},2^{-2},10^{-1})$ 0.0116	**0.0777±0.0511** $(10^{0},2^{-2},10^{-2})$ 0.0016	**0.0777±0.0511** $(10^{0},2^{-2},10^{-2})$ 0.0025	**0.0777±0.0511** $(10^{0},2^{-2},10^{-2})$ 0.0022
Servo (167 X 4)	**0.0767±0.0530** $(10^{2},2^{-1},10^{-3})$ 0.0436	0.0797±0.0417 $(10^{3},2^{0},10^{-3})$ 0.0530	0.0797±0.0417 $(10^{5},2^{0},10^{-3})$ 0.0566	0.0797±0.0271 $(10^{4},2^{-1},10^{-3})$ 0.0069	0.0787±0.0378 $(10^{4},2^{0},10^{-3})$ 0.0710	0.0787±0.0378 $(10^{4},2^{0},10^{-3})$ 0.1684
Triazines (186 X 58)	0.1685±0.0414 $(10^{0},2^{-3},10^{-1})$ 0.0539	**0.1678±0.0292** $(10^{-5},2^{-4},10^{-2})$ 0.0420	**0.1678±0.0296** $(10^{0},2^{-4},10^{-1})$ 0.0421	0.1679±0.0324 $(10^{1},2^{-4},10^{-1})$ 0.0091	0.1679±0.0324 $(10^{1},2^{-4},10^{-1})$ 0.0170	0.1679±0.0324 $(10^{1},2^{-4},10^{-1})$ 0.0172
Wisconsin B.C. (194 X 34)	0.1787±0.0563 $(10^{0},2^{-4},10^{-1})$ 0.0636	0.1797±0.0546 $(10^{-1},2^{-5},10^{-3})$ 0.0434	0.1797±0.0546 $(10^{-1},2^{-5},10^{-3})$ 0.0432	**0.1778±0.0551** $(10^{-5},2^{-5},10^{-3})$ 0.0126	**0.1778±0.0551** $(10^{-5},2^{-5},10^{-3})$ 0.0133	**0.1778±0.0551** $(10^{-5},2^{-5},10^{-3})$ 0.0126
Boston (506 X 13)	0.0780±0.0257 $(10^{2},2^{-5},10^{-2})$ 0.7969	0.0769±0.0168 $(10^{-1},2^{-3},10^{-3})$ 0.3755	0.0769±0.0168 $(10^{-1},2^{-3},10^{-3})$ 0.3719	**0.0768±0.0201** $(10^{0},2^{-3},10^{-3})$ 0.1082	**0.0768±0.0201** $(10^{0},2^{-3},10^{-3})$ 0.2489	0.0769±0.0154 $(10^{0},2^{-3},10^{-3})$ 0.2517
Forest fires (517 X 12)	0.0377±0.0479 $(10^{-5},2^{-5},10^{-3})$ 0.8668	**0.0375±0.0470** $(10^{-1},2^{-5},10^{-1})$ 0.4619	0.0376±0.0469 $(10^{-1},2^{-5},10^{-3})$ 0.4526	0.0391±0.0463 $(10^{-5},2^{-5},10^{-3})$ 0.1111	0.0399±0.0455 $(10^{-5},2^{-5},10^{-3})$ 0.1525	0.0391±0.0463 $(10^{-5},2^{-5},10^{-3})$ 0.1571
ConcreteCS (1030 X 8)	0.0792±0.0056 $(10^{1},2^{-1},10^{-2})$ 5.5429	0.0869±0.0068 $(10^{-1},2^{-1},10^{-1})$ 2.2415	0.0868±0.0077 $(10^{-1},2^{-1},10^{-3})$ 2.2340	**0.0788±0.0065** $(10^{4},2^{-1},10^{-1})$ 0.7944	0.0865±0.0068 $(10^{0},2^{-1},10^{-3})$ 1.7631	0.0865±0.0068 $(10^{0},2^{-1},10^{-3})$ 1.8199
Wine quality red (1599 X 11)	0.1278±0.0064 $(10^{0},2^{0},10^{-2})$ 17.966	0.1277±0.0083 $(10^{-1},2^{-3},10^{-3})$ 6.9404	0.1276±0.0071 $(10^{-1},2^{-3},10^{-3})$ 6.8864	0.1275±0.0080 $(10^{-5},2^{-3},10^{-3})$ 2.5226	0.1274±0.0100 $(10^{-5},2^{-3},10^{-3})$ 3.9555	**0.1273±0.0082** $(10^{-5},2^{-3},10^{-3})$ 4.1435
Concrete Slump (103 X 10)	0.0232±0.0133 $(10^{3},2^{-5},10^{-3})$ 0.0174	0.0366±0.0134 $(10^{3},2^{-1},10^{-3})$ 0.0220	0.0367±0.0134 $(10^{3},2^{-1},10^{-3})$ 0.0268	**0.0224±0.0065** $(10^{5},2^{-3},10^{-3})$ 0.0023	0.0357±0.0158 $(10^{5},2^{-1},10^{-3})$ 0.0077	0.0357±0.0158 $(10^{5},2^{-1},10^{-3})$ 0.0081
Auto price (159 X 15)	0.0817±0.0277 $(10^{1},2^{-4},10^{-2})$ 0.0417	0.0844±0.0241 $(10^{2},2^{-5},10^{-1})$ 0.0373	0.0844±0.0241 $(10^{1},2^{-5},10^{-3})$ 0.0428	**0.0804±0.0189** $(10^{2},2^{-5},10^{-3})$ 0.0057	0.0843±0.0261 $(10^{5},2^{-5},10^{-3})$ 0.0431	0.0843±0.0261 $(10^{5},2^{-5},10^{-3})$ 0.0427
Flexible robotic arm (1019 X 9)	**0.0143±0.0006** $(10^{5},2^{-5},10^{-2})$ 6.1946	0.0248±0.0028 $(10^{3},2^{0},10^{-3})$ 3.3086	0.0249±0.0028 $(10^{4},2^{0},10^{-3})$ 3.2703	0.0152±0.0010 $(10^{5},2^{-1},10^{-3})$ 0.7808	0.0246±0.0034 $(10^{5},2^{-2},10^{-2})$ 5.2768	0.0241±0.0034 $(10^{5},2^{-2},10^{-2})$ 5.4293

Pollution (60 X 15)	**0.1085±0.0461** $(10^{0},2^{-3},10^{-2})$ 0.0085	0.1113±0.0405 $(10^{-2},2^{-5},10^{-1})$ 0.0316	0.1111±0.0365 $(10^{-1},2^{-5},10^{-1})$ 0.0102	0.1113±0.0423 $(10^{-5},2^{-5},10^{-3})$ 0.0007	0.1113±0.0423 $(10^{-5},2^{-5},10^{-3})$ 0.0008	0.1113±0.0423 $(10^{-5},2^{-5},10^{-3})$ 0.0008
NO2 (500 X 7)	0.0979±0.0124 $(10^{0},2^{0},10^{-2})$ 0.6977	**0.0972±0.0123** $(10^{-2},2^{-1},10^{-1})$ 0.3521	**0.0972±0.0123** $(10^{-2},2^{-1},10^{-3})$ 0.3294	**0.0972±0.0123** $(10^{-5},2^{-1},10^{-3})$ 0.1039	**0.0972±0.0123** $(10^{-5},2^{-1},10^{-3})$ 0.1716	**0.0972±0.0123** $(10^{-5},2^{-1},10^{-3})$ 0.1713
Bodyfat (252 X 14)	**0.0151±0.0228** $(10^{2},2^{-5},10^{-3})$ 0.1246	0.0180±0.0215 $(10^{-1},2^{-4},10^{-1})$ 0.0844	0.0182±0.0215 $(10^{-1},2^{-4},10^{-3})$ 0.0823	0.0204±0.0203 $(10^{-5},2^{-4},10^{-3})$ 0.0154	0.0204±0.0203 $(10^{-5},2^{-4},10^{-3})$ 0.0182	0.0208±0.0204 $(10^{-5},2^{-4},10^{-3})$ 0.0178
Balloon (2001 X 1)	0.0449±0.0025 $(10^{0},2^{0},10^{-1})$ 33.170	0.0452±0.0020 $(10^{0},2^{-2},10^{-1})$ 20.365	0.0452±0.0020 $(10^{0},2^{-2},10^{-3})$ 20.224	**0.0448±0.0040** $(10^{3},2^{-3},10^{-3})$ 5.0500	**0.0448±0.0040** $(10^{2},2^{-2},10^{-3})$ 15.652	**0.0448±0.0040** $(10^{2},2^{-2},10^{-3})$ 16.023
Quake (2178 X 3)	0.1751±0.0161 $(10^{4},2^{-1},10^{-1})$ 55.230	**0.1718±0.0096** $(10^{-5},2^{-5},10^{-1})$ 16.964	**0.1718±0.0097** $(10^{-2},2^{-5},10^{-3})$ 15.641	**0.1718±0.0096** $(10^{-1},2^{-5},10^{-3})$ 6.1877	**0.1718±0.0096** $(10^{-1},2^{-5},10^{-3})$ 12.030	**0.1718±0.0096** $(10^{-1},2^{-5},10^{-3})$ 12.441
Motorcycle (133 X 1)	0.1143±0.0246 $(10^{2},2^{5},10^{-3})$ 0.0254	0.1104±0.0224 $(10^{0},2^{5},10^{-1})$ 0.0249	0.1096±0.0222 $(10^{0},2^{5},10^{-2})$ 0.0313	0.1095±0.0217 $(10^{1},2^{5},10^{-3})$ 0.0042	**0.1092±0.0189** $(10^{1},2^{5},10^{-3})$ 0.0097	**0.1092±0.0189** $(10^{1},2^{5},10^{-3})$ 0.0100
Demo (2048 X 4)	0.0885±0.0108 $(10^{0},2^{4},10^{-2})$ 42.370	**0.0873±0.0110** $(10^{-2},2^{1},10^{-1})$ 13.610	**0.0873±0.0110** $(10^{-2},2^{1},10^{-2})$ 13.846	**0.0873±0.0082** $(10^{-1},2^{1},10^{-3})$ 5.2906	**0.0873±0.0082** $(10^{-1},2^{1},10^{-3})$ 10.403	**0.0873±0.0082** $(10^{-1},2^{1},10^{-3})$ 10.719
Sunspots (290 X 5)	0.0727±0.0100 $(10^{1},2^{-1},10^{-2})$ 0.1722	0.0715±0.0124 $(10^{-1},2^{0},10^{-1})$ 0.1037	0.0710±0.0076 $(10^{-2},2^{1},10^{-2})$ 0.0998	0.0708±0.0090 $(10^{1},2^{1},10^{-1})$ 0.0300	**0.0707±0.0104** $(10^{1},2^{1},10^{-1})$ 0.0526	**0.0707±0.0104** $(10^{1},2^{1},10^{-1})$ 0.0568
IBM (750 X 5)	0.0272±0.0025 $(10^{2},2^{-5},10^{-2})$ 2.2962	0.0270±0.0023 $(10^{-1},2^{-2},10^{-1})$ 1.0976	0.0270±0.0023 $(10^{-1},2^{-2},10^{-3})$ 1.0781	0.0269±0.0038 $(10^{-1},2^{-2},10^{-3})$ 0.3064	**0.0267±0.0038** $(10^{-1} ,2^{-2},10^{-3})$ 0.6273	**0.0267±0.0038** $(10^{-1},2^{-2},10^{-3})$ 0.6545
SNP500 (750 X 5)	**0.0220±0.0033** $(10^{1},2^{-5},10^{-3})$ 2.3142	0.0231±0.0052 $(10^{0},2^{-5},10^{-1})$ 1.3837	0.0221±0.0049 $(10^{0},2^{-1},10^{-2})$ 1.2905	0.0224±0.0051 $(10^{5},2^{-5},10^{-3})$ 0.3321	0.0229±0.0052 $(10^{1},2^{-5},10^{-3})$ 0.8775	0.0229±0.0052 $(10^{1},2^{-5},10^{-3})$ 0.9510
Citigroup (750 X 5)	**0.0147±0.0024** $(10^{1},2^{-2},10^{-3})$ 2.2938	0.0149±0.0029 $(10^{0},2^{-2},10^{-3})$ 1.2793	0.0149±0.0026 $(10^{-1},2^{-2},10^{-3})$ 1.1395	0.0149±0.0028 $(10^{4},2^{-4},10^{-3})$ 0.3368	0.0149±0.0026 $(10^{1},2^{-2},10^{-3})$ 0.9140	0.0149±0.0026 $(10^{1},2^{-2},10^{-3})$ 0.9269
Intel (750 X 5)	0.0292±0.0044 $(10^{3},2^{-5},10^{-3})$ 2.3469	**0.0290±0.0042** $(10^{-2},2^{-1},10^{-1})$ 0.9380	**0.0290±0.0042** $(10^{-2},2^{-1},10^{-3})$ 0.9223	**0.0290±0.0042** $(10^{-1},2^{-1},10^{-3})$ 0.3077	**0.0290±0.0042** $(10^{-1},2^{-1},10^{-3})$ 0.6279	**0.0290±0.0042** $(10^{-1},2^{-1},10^{-3})$ 0.6471
Microsoft (750 X 5)	**0.0277±0.0056** $(10^{2},2^{-5},10^{-3})$ 2.2596	0.0283±0.0059 $(10^{0},2^{-5},10^{-3})$ 1.3622	0.0283±0.0059 $(10^{0},2^{-5},10^{-3})$ 1.3571	0.0284±0.0060 $(10^{2},2^{-5},10^{-3})$ 0.3321	0.0285±0.0059 $(10^{1},2^{-5},10^{-3})$ 0.8755	0.0286±0.0052 $(10^{1},2^{-5},10^{-3})$ 0.9003
RedHat (750 X 5)	**0.0256±0.0070** $(10^{1},2^{-1},10^{-3})$ 2.3019	0.0259±0.0075 $(10^{-1},2^{-5},10^{-1})$ 1.0486	0.0258±0.0052 $(10^{-1},2^{-5},10^{-3})$ 1.0335	**0.0256±0.0078** $(10^{0},2^{-4},10^{-3})$ 0.3116	**0.0256±0.0078** $(10^{0},2^{-4},10^{-3})$ 0.7525	**0.0256±0.0078** $(10^{0},2^{-4},10^{-3})$ 0.7725

Table 5.6: Average ranks of SVR, TSVR, TWSVR, FLTWSVR, NLTWSVR and GLTWSVR with Gaussian kernel.

Dataset	**SVR**	**TSVR**	**TWSVR**	**FLTWSVR**	**NLTWSVR**	**GLTWSVR**
Hydraulic actuator	6	4	5	2.5	1	2.5
Gas furnace	1	3.5	3.5	2	6	5
Pyrim	6	4.5	4.5	2	2	2
Servo	1	5	5	5	2.5	2.5
Triazines	6	1.5	1.5	4	4	4
Wisconsin B.C.	4	5.5	5.5	2	2	2
Boston	6	4	4	1.5	1.5	4
Forest fires	3	1	2	4.5	6	4.5
ConcreteCS	2	6	5	1	3.5	3.5
Wine quality red	6	5	4	3	2	1
Concrete Slump	2	5	6	1	3.5	3.5
Auto price	2	5.5	5.5	1	3.5	3.5
Flexible robotic arm	1	5	6	2	4	3
Pollution	1	4.5	2	4.5	4.5	4.5
NO2	6	3	3	3	3	3
Bodyfat	1	2	3	4.5	4.5	6
Balloon	4	5.5	5.5	2	2	2
Quake	6	3	3	3	3	3
Motorcycle	6	5	4	3	1.5	1.5
Demo	6	3	3	3	3	3
Sunspots	6	5	4	3	1.5	1.5
IBM	6	4.5	4.5	3	1.5	1.5
SNP500	1	6	2	3	4.5	4.5
Citigroup	1	4	4	4	4	4
Intel	6	3	3	3	3	3
Microsoft	1	2.5	2.5	4	5	6
Redhat	2.5	6	5	2.5	2.5	2.5
Average Rank	3.6851	4.1666	3.9259	**2.8518**	3.1481	3.2222

Chapter 6

Conclusion and Future Research

6.1 Conclusion

A new iterative Lagrangian twin support vector regression based on twin support vector machine (LTWSVR) for the twin support vector machine based regression (TWSVR) is proposed. This leads to the minimization problem having strongly convex objective functions with non-negativity constraints. LTWSVR requires at the outset the inverse of a matrix but this can be expressed as matrix subtraction of identity matrix by a scalar multiple of the inverse of a positive semi-definite matrix. Further it is proposed to solve this problem by simple iterative methods: functional iterative method (FLTWSVR), Newton method (NLTWSVR) and Generalized derivative approach (GLTWSVR). Our formulation has the advantage that it does not need any optimization tools of linear or quadratic programming solvers. Numerical experiments were performed on a number of interesting synthetic and real-world benchmark datasets. The results obtained show similar or better generalization performance with smaller computation time in comparison with SVR, TSVR and TWSVR.

6.2 Future Research

Future work will include the study of implicit Lagrangian formulation (Mangasarian and Solodov, 1993) for the dual TWSVR problem and its applications. There is also a room for study of smoothing approach for solving Lagrangian TWSVR.

References

D. Achlioptas, F. McSherry, and B. Sch¨olkopf. Sampling techniques for kernel methods, In Advances in Neural Information Processing Systems 14, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds., MIT Press, Cambridge, MA, 2002.

M.A. Aizerman, E.M. Braverman, and L.I. Rozono'er. Theoretical foundations of the potential function method in pattern recognition learning. Automation and Remote Control 25, pp.821–837, 1964.

S. Balasundaram, M. Tanveer, On Lagrangian twin support vector regression, Neural Computing & Applications, 22(1), pp. 257-267, 2013

S. Balasundaram & R. Singh. On finite Newton method for support vector regression. Neural Computing & Applications, 19(7), pp. 967–977, 2010.

S. Ben-David, M. Lindenbaum. Learning distributions by their density levels: a paradigm for learning without a teacher, Journal of Computer and System Sciences, 55, pp. 171–182, 1997.

J. Bi & K.P. Bennett. A geometric approach to support vector regression, Neurocomputing, vol. 55, pp. 79-108, 2003.

B.E. Boser, I.M. Guyon and V.N. Vapnik. A training algorithm for optimal margin classifiers. In proceedings of the Annual Conference on Computational Learning Theory, D. Haussler, Ed., ACM Press, Pittsburgh, PA, pp. 144–152, 1992.

G.E.P. Box and G.M. Jenkins. Time series analysis: Forecasting and control (Holden-Day, San Francisco, CA), 1976.

C.J.C. Burges, B. Sch¨olkopf. Improving the accuracy and speed of support vector learning machines, In M. Mozer, M. Jordan, and T. Petsche, editors, Advances in Neural Information Processing Systems 9, pp. 375–381, MIT Press, Cambridge, MA, 1997.

C. Cortes, V.N. Vapnik,. Support vector networks. Machine Learning, 20, pp. 273–297, 1995.

N. Cristianini, J. Shawe-Taylor. An introduction to support vector machines and other kernel-based learning methods, Cambridge University Press, Cambridge, 2000.

Delve. Data for Evaluating Learning in Valid Experiments, http://www.cs.toronto.edu/~delve/data, 2005.

J. Demsar. Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research, 7, pp. 1–30, 2006.

R. Eubank. Nonparametric Regression and Spline Smoothing, Marcel Dekker, New York, NY, 1999.

F. Facchinei. Minimization of SC^1 functions and the Maratos effect. Operations Research Letters, 17, pp. 131–137, 1995.

S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations, Journal of Machine Learning Research, 2:243–264, Dec 2001.

G. Fung, & O. L. Mangasarian. Finite Newton method for Lagrangian support vector machine, Neurocomputing, 55, pp. 39–55, 2003.

A. Gretton, A. Doucet, R. Herbrich, P.J.W. Rayner, B. Schölkopf. Support vector regression for black-box system identification, In proceedings of the 11th IEEE Workshop on Statistical Signal Processing, 2001.

S. R. Gunn. Support Vector Machines for Classification and Regression, Technical Report, 1998.

J.-B. Hiriart-Urruty, J. J. Strodiot, & V. H. Nguyen. Generalized hessian matrix and second-order optimality conditions for problems with C^{L1} data, Applied Mathematics and Optimization, 11, pp. 43–56, 1984.

Jayadeva, R. Khemchandai, and S. Chandra. Twin support vector machines for pattern classification, IEEE Transaction on Pattern Analysis and Machine Intelligence (TPAMI), vol. 29, pp. 905-910, 2007.

T. Joachims. Making large-scale SVM learning practical. In Advances in Kernel Methods—Support Vector Learning, B. Sch¨olkopf, C.J.C. Burges, and A.J. Smola, Eds., MIT Press, Cambridge, MA, pp. 169–184, 1999.

T. Joachims. Learning to classify text using support vector machines, Kluwer Academic Publishers, London, 2002.

W. Karush. Minima of functions of several variables with inequalities as side constraints. Masters thesis. Department of Mathematics, University of Chicago, 1939.

L. Kaufman. Solving the quadratic programming problem arising in support vector classification. In Advances in Kernel Methods—Support Vector Learning, B. Sch¨olkopf, C.J.C. Burges and A.J. Smola, Eds., MIT Press, Cambridge, MA, pp. 147–168, 1999.

V. Kecman. Learning and Soft Computing: Support Vector Machines, Neural Networks, and Fuzzy Logic Models, MIT Press, Cambridge 2001.

R. Khemchandani, K. Goyal, S. Chandra. Twin support vector machine based Regression, 2015 Eigth International Conference On Advances in Pattern Recognition(ICAPR), pp. 1-6, Jan 2015.

K. J. Kim. Financial time series forecasting using support vector machines. Neurocomputing, 55(1/2), pp. 307–319, 2003.

H.W. Kuhn, A.W. Tucker, 1951. Nonlinear programming. In Proceedings of 2nd Berkeley Symposium on Mathematical Statistics and Probabilistics, University of California Press, Berkeley, pp. 481–492, 1951.

Y. J. Lee, W. F. Heisch and C. M. Huang. ε - SSVR: A smooth support vector machine for ε -insensitive regression,” IEEE Trans. Knowl. Data Eng., vol. 17, pp. 678–685, 2005.

O.L. Mangasarian. A finite Newton method for classification problems, Technical Report 01-11,Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, December 2001. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/01-11.ps. Optimization Methods and Software, vol. 17, pp.913–929, 2002.

O.L. Mangasarian, MV. Solodov. Nonlinear complementarity as unconstrained and constrained minimization. Math Program B 62:277–297, 1993.

O.L. Mangasarian and E.W. Wild. Multisurface Proximal Support Vector Classification via Generalized Eigenvalues, IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 28, no. 1, pp. 69-74, Jan. 2006.

O.L. Mangasarian and D. Musicant. Lagrangian support vector machines. Journal of Machine Learning Research (JMLR), vol 1, pp. 161–177, 2001.

S. Mukherjee, E. Osuna, F. Girosi, Nonlinear prediction of chaotic time series using support vectormachines, In proceedings of the IEEE Workshop on Neural Networks for Signal Processing, Amelia Island, FL, USA, pp. 511–520, 1997.

P.M. Murphy & Aha, D.W. UCI machine learning repository. http://www.ics.uci.edu/~mlearn/ MLRepository.html, 1992.

E. Osuna. R. Freund, and E. Cirosi. Training support vector machines: an application to face detection, In Proceedings of Computer Vision and Pattern Recognition, pages 130-136, IEEE Computer Society Press, 1997.

C. P. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In International Conference on Computer Vision, Bombay, India, January 1998.

X. Peng, TSVR: an efficient twin support vector machine for regression, Neural Networks 23(3). 365–372, 2010.

J. Platt. Fast training of support vector machines using sequential minimal optimization. In Advances in Kernel Methods—Support Vector Learning, B. Sch¨olkopf, C.J.C. Burge, and A.J. Smola, Eds., pp. 185-208, MIT Press, Cambridge, MA,, 1999.

B. Ribeiro Kernelized based functions with Minkovsky’s norm for SVM regression. In: Proceedings of the international joint conference on neural networks, 2002, IEEE press, pp 2198–2203, 2002.

M. Rychetsky, S. Ortmann, and M. Glesner. Support vector approaches for engine knock detection, Neural Networks, vol. 2, pp. 969–974, 1999.

B. Scholkopf , C.J.C. Burges, and A.J. Smola. Advances in Kernel Methods—Support Vector Learning, MIT Press, Cambridge, MA, 1999.

B. Scholkopf and A. Smola. Learning with Kernels. MIT Press, Cambridge, MA, 2002.

S.K. Shevade, S.S. Keerthi, C. Bhattacharyya et al. Improvements to the SMO algorithm for SVM regression. IEEE Transactions on Neural Networks, 11(5), 1188-1193, 2000.

B.W. Silverman . Some aspects of the spline smoothing approach to non-parametric curve fitting. Journal of the Royal Statistical Society Series B, 47 (1) (1985), pp. 1–52, 1985.

J. Sjöberg, Q. Zhang, L. Ljung, A. Beneviste, B. Delyon, P.Y. Glorennec, H. Hjalmarsson, A. Juditsky. Nonlinear black-box modeling in system identification: a unified overview Automatica, 31, pp. 1691–1724, 1995.

L.G.M. Souza, G.A Barreto. Multiple local ARX modeling for system identification using the self-organizing map. Proc. of European Symp. on Artificial Neural Networks – Computational Intelligence and Machine Learning. Bruges, 2010.

D. Stoneking. Improving the manufacturability of electronic designs. IEEE Spectrum, 36(6), 70–76, 1999.

J.A.K. Suykens, J.Vandewalle. Least squares support vector machine classifiers. Neural Processing Letters,Vol.9, No.3, pp.293-300, 1999.

J.A.K. Suykens, L. Lukas, P. Van Dooren, B. De Moor, J. Vandewalle. Least squares support vector machine classifiers: a large scale algorithm, European Conference on Circuit Theory and Design, (ECCTD'99), Stresa Italy, pp.839-842, Aug 1999.

L. Tarassenko, P. Hayton,N. Cerneaz, M. Brady. Novelty detection for the identification of masses in mammograms, Proceedings fourth IEE international conference on artificial neural networks, Cambridge, pp. 442–447, 1995.

F.E.H. Tay and L. J. Cao. Application of support vector machines in financial time series forecasting, Omega, vol. 29, pp. 309–317, 2001.

I.W. Tsang, J.T. Kwok, and P.M. Cheung. Very large SVM training using core vector machines. In proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, Barbados, Jan 2005.

P. Vlachos. Statlib datasets archive, URL http://lib.stat.cmu.edu/datasets/, 2005.

V. N. Vapnik. The nature of statistical learning theory, 2nd Edition, Springer, New York, 2000.