

**Newton's method for Implicit Lagrangian Twin  
Support Vector Regression (LTSVR)**

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

**MASTER OF TECHNOLOGY  
IN  
COMPUTER SCIENCE AND TECHNOLOGY**

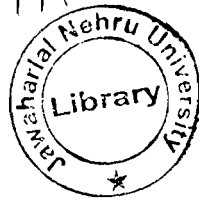
**BY  
DEEPAK GUPTA**



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

**JULY 2011**

TH-20618



THESES/DISSERTATION



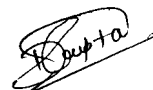
TH20618

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

DECLARATION

This is to certify that the dissertation entitled “*Newton’s method for Implicit Lagrangian Twin Support Vector Regression (LTSVR)*” 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 and 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.



Deepak Gupta

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

CERTIFICATE

This is to certify that the dissertation entitled “*Newton’s method for Implicit Lagrangian Twin Support Vector Regression (LTSVR)*” being submitted by *Mr. Deepak Gupta* to the *School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi* in partial fulfillment of requirements for the award of the degree of *Master of Technology in Computer Science and Technology*, is a record of bonafide work carried out by him under the supervision of *Prof. S. Balasundaram*.



24/7/11

Prof. S. Balasundaram  
(Supervisor)  
SC&SS, JNU, New Delhi



Dean  
SC&SS  
JNU, New Delhi

## ACKNOWLEDGEMENT

I would like to express my honest gratitude to my supervisor, Prof. S. Balasundaram of the School of Computer and Systems Science for his outstanding help, support, guidance and efforts during my research. Successful completion of the present research effort has not been possible without him, who not only served as my guide but also encouraged and challenged me throughout my research program. He patiently guided me through the dissertation process, never accepting less than my best efforts.

I would also like to choose this opportunity to express my respect to my parents and my family who always motivated me to go forward and complete the task at hand and finally, to my friends, specially Kapil, Tanveer, Shina, Asit, Ateet, Govind and colleagues who kept me sane the long hours of work.

Deepak Gupta

*DEDICATED TO*

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

# ABSTRACT

In this work, an implicit Lagrangian for the dual twin support vector regression is proposed. Our formulation leads to determining a pair of non-parallel  $\varepsilon$ -insensitive down- and up- bound functions for the unknown regressor by constructing a pair of unconstrained minimization problems in which each of them will be of smaller in size in comparison to a single large one as in the standard support vector regression (SVR) and solving them using Newton method. Numerical experiments were performed on a number of interesting synthetic and real-world benchmark datasets and their results were compared with SVR and twin support vector regression (TSVR). Similar or better generalization performance of the proposed method clearly illustrates its effectiveness and applicability.

## Table of Contents

CHAPTER 1. Support Vector Machines for Classification and Regression Problems...	1
1.1 Introduction.....	1
1.2 Linear SVM.....	3
1.3 Imperfect separation (Soft Margin SVM).....	5
1.4 Non-Linear SVM.....	6
1.5 Linear SVR.....	8
1.6 Non-Linear SVR.....	9
CHAPTER 2. Twin Support Vector Regression.....	11
2.1 Linear TSVR.....	11
2.2 Kernel TSVR.....	15
CHAPTER 3. Implicit Lagrangian Twin Support Vector Regression (LTSVR).....	17
3.1 Linear LTSVR.....	17
3.2 Non-Linear LTSVR.....	21
3.3 Newton Method.....	24
CHAPTER 4. Experimental Result.....	26
4.1 Introduction.....	26
4.2 Experiments on synthetic data sets.....	27
4.3 Experiments on real-world benchmark data sets .....	32
CONCLUSION AND FUTURE WORK.....	37
APPENDIX	
MATLAB code for implicit LTSVR using Newton's method	
REFERENCES	



## Table of Figures :

Figure 1: Linear Separating plane.....	3
Figure 2: Soft Margin Classifier.....	5
Figure 3: Mapping into a higher dimension space.....	7
Figure 4: Graphical representation of the linear regression function and the epsilon bond..	9
Figure 5: The geometric interpretation of TSVR.....	12
Figure 6a: Results of approximation of $\frac{4}{ x +2} + \cos(2x) + \sin(3x)$ with our proposed method, SVR and TSVR on noisy input samples for uniform noises over the interval $[-0.2, 0.2]$ .....	30
Figure 6b: Results of approximation of $\frac{4}{ x +2} + \cos(2x) + \sin(3x)$ with our proposed method, SVR and TSVR on noisy input samples for Gaussian noises with mean zero and standard deviation 0.2.....	30
Figure 7a: Results of approximation of $\frac{4}{ x +2} + \cos(2x)$ with our proposed method, SVR and TSVR on noisy input samples for uniform noises over the interval $[-0.2, 0.2]$ .....	31
Figure 7b: Results of approximation of $\frac{4}{ x +2} + \cos(2x)$ with our proposed method, SVR and TSVR on noisy input samples for Gaussian noises with mean zero and standard deviation 0.2.....	31
Figure 8a: Results of comparison for gas furnace data of Box-Jenkins for Prediction over the training set.....	36
Figure 8b: Results of comparison for gas furnace data of Box-Jenkins for Prediction over the test set.....	36

## Tables :

Table 1. Functions used for generating synthetic datasets.....	29
Table 2. Performance comparison of our proposed method with SVR and TSVR on synthetic datasets for uniformly distributed and Gaussian noises.....	29
Table 3. Performance comparison of our proposed method with SVR and TSVR on real world datasets. RMSE was used for comparison.....	35

# CHAPTER 1

## Support Vector Machines for Classification and Regression problems

### 1.1 Introduction

Support vector machines (SVMs) are supervised learning methods used for classification and regression problems proposed by Vapnik (Vapnik, 2000). Initially, SVMs were developed for classification problems and later extended for regression estimation problems. SVM is a powerful tool for data classification. Face detection (Osuna et al., 1997), Gene prediction (Tong et al., 2005), Drug discovery (Demiriz et al., 2001), Stock exchange prediction (Bao et al., 2005), Time series forecasting (Brockwell & Davis, 2002; Cao, 2003; Mukherjee et al., 1997; Muller et al., 1999) are numerous classes of problems which fall under classification or regression problem domain. Generally machine learning methods determine the nonlinear function relationship between dependent and independent variables by learning from the input samples. One of the machine learning method namely Artificial Neural Networks (ANNs) trained with back propagation (BP) (Lawrence & Jeanette, 1994) is very popular which have many advantages like better approximation capabilities. However, it has certain disadvantages like selection of the number of hidden layer neurons, slow convergence, imprecise learning rate etc. Recently a novel machine learning method called Support Vector Machine (SVM) (Burges, 1998; Cristianini & Taylor, 2000; Vapnik, 2000), proposed by Vapnik (Vapnik, 2000) applied to classification and regression estimation problems. For a given a set of training examples with class labels  $\{ +1 \text{ or } -1 \}$  an SVM training algorithm builds a model that predicts whether a new example falls in either positive or negative category.

Initially SVMs were developed for classification problems and later extended for regression estimation problems. SVM is a powerful tool for data classification. In SVM, classification is achieved by constructing a linear or non-linear separating surface in the input space of the dataset or in a higher dimensional feature space. SVM formulation leads to the solution of a quadratic programming problem (QPP) with linear inequality constraints. Combined with the better generalization ability SVM becomes a powerful paradigm of choice for pattern classification.

The objective of regression problem lies in determining the underlying mathematical relationship between the given input observations and their output values. Vapnik (Vapnik, 2000) introduced  $\varepsilon$ -insensitive error loss function to SVM methods which has got successful application to regression problems (Mukherjee et al., 1997; Tay & Cao, 2001). SVR has been applied to many problems of practical importance such as time series prediction (Tay et al., 2001; Muller et al., 1999, Mukherjee et al., 1997), Drug discovery (Demiriz et al., 2001), Civil Engineering (Dibike et al., 2000).

We assume all vectors are considered as column vectors. For any two vectors  $x, y$  in  $R^n$ , the inner product of the two vectors will be denoted by  $x'y$  where  $x'$  is the transpose of vector. When  $x$  is orthogonal to  $y$ , we write  $x \perp y$ . For a given vector  $x = (x_1, x_2, \dots, x_n)'$  in  $R^n$ , the plus function  $x_+$  is defined as:  $(x_+)_i = \max\{0, x_i\}$  for  $i = 1, 2, \dots, n$  and the step function  $x_*$  will be defined as:  $(x_*)_i = 1$  if  $x_i > 0$ ,  $(x_*)_i = 0$  if  $x_i < 0$  and  $(x_*)_i = 0.5$  if  $x_i = 0$ . The 2-norm of a vector  $x$  and a matrix  $Q$  will be denoted by  $\|x\|$  and  $\|Q\|$  respectively. The column vector of ones of dimension  $m$  is denoted by  $e$ . For a matrix  $A \in R^{m \times n}$ ,  $A_i$  is the  $i^{\text{th}}$  row of  $A$  which is a row vector in  $R^n$ . For  $A \in R^{m \times n}$  and  $B \in R^{n \times k}$  be two matrices, the kernel  $K(A, B)$  maps  $R^{m \times n} \times R^{n \times k}$  into  $R^{m \times k}$ . If  $f$  is a real valued function of the variable  $x = (x_1, x_2, \dots, x_n)'$  in  $R^n$  then we denote the gradient of  $f$  by  $\nabla f$  and it is a vector in  $R^n$  defined by:

$$\nabla f = \left( \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n} \right)'. \text{ Also } \nabla^2 f = \left( \frac{\partial^2 f}{\partial x_i \partial x_j} \right)_{i,j=1,2,\dots,n}$$
 is the hessian matrix of  $f$ .

Assume that a set of  $m$  input observation  $\{(x_i, y_i)\}_{i=1,2,\dots,m}$  where  $x_i = (x_{i1}, x_{i2}, \dots, x_{in})' \in R^n$  and  $y_i = \{+1, -1\}$  be given. The problem is in determining a classifier  $f(\cdot)$  such that

$$f(x_i) = \begin{cases} +1 & \text{for } (x_i, y_i) \text{ with } y_i = +1 \\ -1 & \text{for } (x_i, y_i) \text{ with } y_i = -1 \end{cases} \quad (1.1)$$

and the classifier  $f(\cdot)$  has better generalization ability.

## 1.2 Linear SVM

For the above set of input examples given, the linear SVM determines the separating hyperplane to be the form  $H : x'w + b = 0$  such that the decision function is given by

$$f(x) = \text{sign}(x'w + b) \quad (1.2)$$

where  $w \in R^n$ ,  $b \in R$  are unknowns. The hyperplane  $H : x'w + b = 0$  is constructed so that it has maximum margin between the two classes. For this, two parallel hyperplanes are constructed on both sides of the separating hyperplane having equal distance to the separating hyperplane with the condition that no data points between  $H1$  and  $H2$  and the distance between  $H1$  and  $H2$  is maximized where

$$H1: \quad x'w + b = +1 \quad (1.3)$$

$$H2: \quad x'w + b = -1 \quad (1.4)$$

By normalizing the coefficients by means of applying suitable transformations, if necessary, we can always assume that  $H1$  and  $H2$  will take the above forms. Also by vector geometry, the distance between two hyperplanes, called margin, is computed to be  $\frac{2}{\|w\|}$ , where  $\|\cdot\|$  is the Euclidean norm. See the Figure 1.

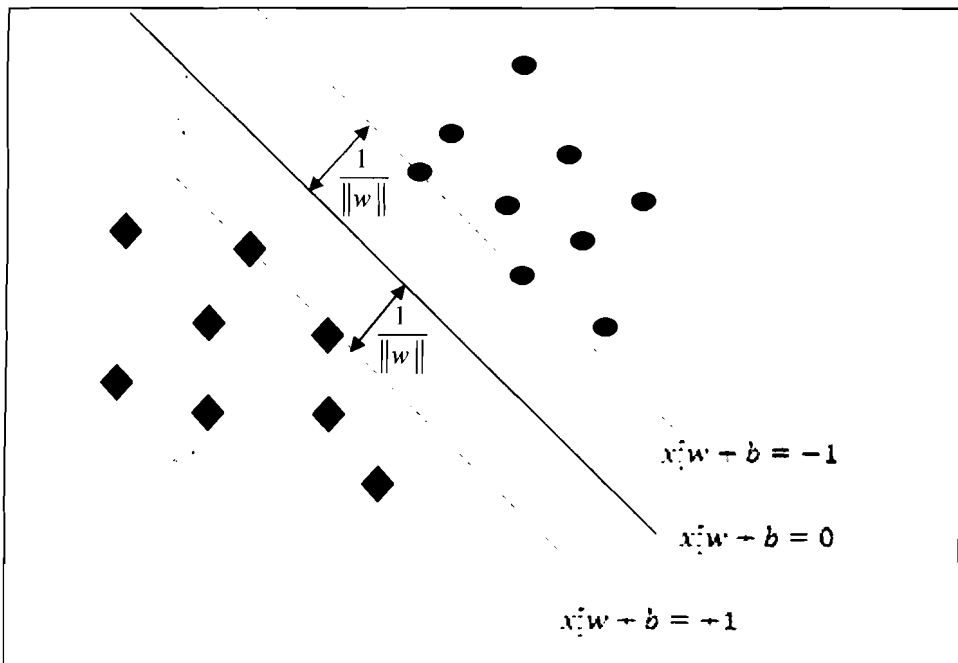


Figure 1: Linear Separating plane

Thus the problem is formulated to be:

$$\max \frac{2}{\|w\|}$$

subject to:  $y_i(x^t w + b) \geq 1 \quad \forall i = 1, \dots, m$

or equivalently:

$$\min \frac{\|w\|^2}{2}$$

subject to  $y_i(x^t w + b) \geq 1 \quad \forall i = 1, \dots, m$  (1.5)

The above primal problem can be formulated into its dual as follows.

Introducing Lagrange multipliers  $\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_m \geq 0$ , we find the following Lagrangian function:

$$L(w, b, \alpha) = \frac{1}{2} w^t w - \sum_{i=1}^m \alpha_i y_i (x^t w + b) + \sum_{i=1}^m \alpha_i$$
 (1.6)

Equating the gradient of  $L(w, b, \alpha)$  with respect to the primal variables  $w$  and  $b$  to zero,

$$\frac{\partial L(w, b, \alpha)}{\partial w} = 0 \quad \text{and} \quad \frac{\partial L(w, b, \alpha)}{\partial b} = 0$$
 (1.7)

we get,

$$w = \sum_{i=1}^m \alpha_i y_i x_i \quad \text{and} \quad \sum_{i=1}^m \alpha_i y_i = 0$$
 (1.8)

Substituting the above conditions into the Lagrangian (1.6), we obtain the following minimization problem in dual variables as:

$$\min \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j (x_i^t x_j) - \sum_{i=1}^m \alpha_i$$

subject to:  $\alpha_i \geq 0 \quad \forall i = 1, \dots, m$  (1.9)

Using the solution  $\alpha_i$  for the above dual problem, the primal variables  $w$  and  $b$  can be

determined. So the classifier function being  $f(x) = \text{sign}(\sum_{i=1}^n \alpha_i y_i (x^t x) + b)$

where  $b = y_i - x_i^t w$  for  $i = \arg \max_j (\alpha_j)$

### 1.3 Imperfect separation (Soft Margin SVM)

In the case of Linear SVM, data points are not allowed to lie between the two hyperplanes. The other direction to extend SVM is to allow for noise, or imperfect separation. That is, we do not strictly enforce there be no data points between  $H1$  and  $H2$ , but we definitely want to penalize the data points that cross the boundaries. In this case penalty  $C$  will be finite. (If  $C = \infty$ , we come back to the original perfect separating case).

For those input data points that lie between the planes as shown in Fig.2, we introduce 'error'  $\xi_i$  (a slack variable) in classification as a penalty.

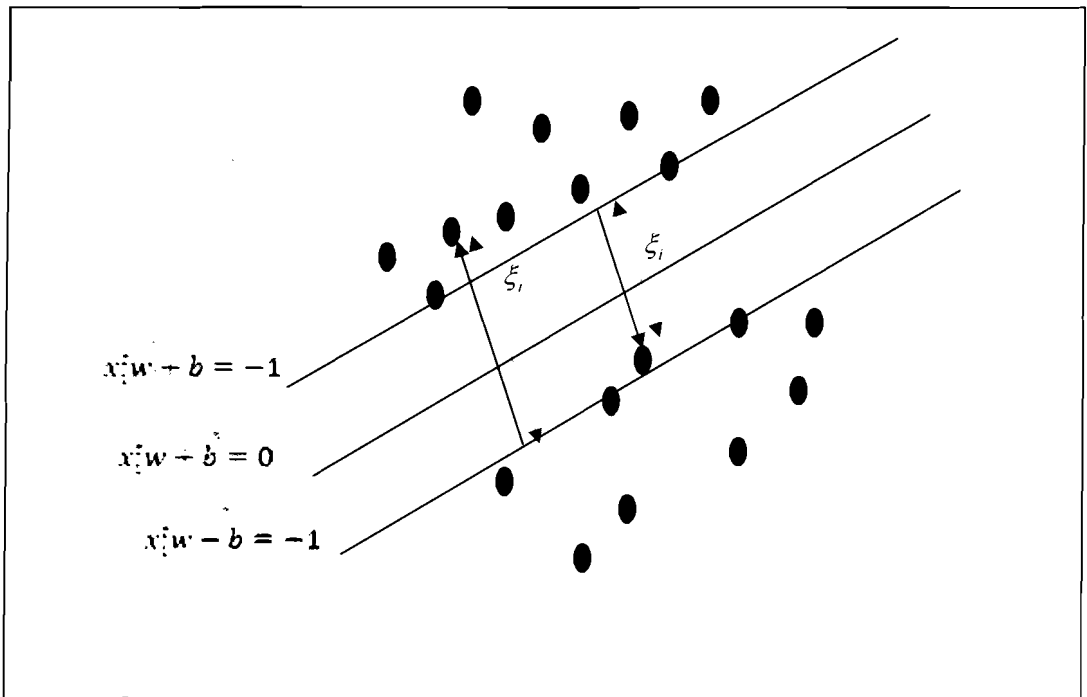


Figure 2: Soft Margin Classifier

With the introduction of penalty, the problem can be reformulated as follows:

$$\begin{aligned}
 x_i'w + b &\geq +1 - \xi_i, \text{ for } y_i = +1, \\
 x_i'w + b &\leq -1 + \xi_i, \text{ for } y_i = -1, \\
 \xi_i &\geq 0, \forall i = 1, \dots, m
 \end{aligned}
 \tag{1.10}$$

and further in this case we will have:

$$\min \frac{1}{2} w'w + C \sum_{i=1}^m \xi_i$$

$$\begin{aligned} \text{subject to: } & y_i(x^t w + b) + \xi_i - 1 \geq 0, \forall i = 1, \dots, m \\ & \xi_i \geq 0, \forall i = 1, \dots, m \end{aligned} \quad (1.11)$$

where  $C > 0$  is a parameter.

By introducing Lagrange multipliers  $\alpha = (\alpha_1, \dots, \alpha_m)'$ ,  $\beta = (\beta_1, \dots, \beta_m)'$   $\alpha_i \geq 0, \beta_i \geq 0 \quad \forall i = 1, \dots, m$ , the Lagrangian function for (1.11) can be defined to be

$$L(w, b, \xi, \alpha, \beta) = \frac{1}{2} w^t w + C \sum_{i=1}^m \xi_i - \sum_{i=1}^m \alpha_i [1 - \xi_i - y_i(x^t w + b)] - \sum_{i=1}^m \beta_i \xi_i \quad (1.12)$$

Making the gradient of  $L(w, b, \xi, \alpha, \beta)$  with respect to the primal variables  $w, b$  and  $\xi_i$  to zero, and proceeding in similar manner as in the previous case, we obtain:

$$w = \sum_{i=1}^m \alpha_i y_i x_i, \quad \sum_{i=1}^m \alpha_i y_i = 0 \quad \text{and} \quad C - \alpha_i - \beta_i = 0 \quad (1.13)$$

Substituting the above conditions in the definition of the Lagrangian function, the dual problem can be obtained and be stated as

$$\begin{aligned} \min & \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j (x_i^t x_j) - \sum_{i=1}^m \alpha_i \\ \text{subject to: } & 0 \leq \alpha_i \leq C \quad \forall i = 1, \dots, m \end{aligned} \quad (1.14)$$

The only difference of (1.14) with linear separable case (1.9) is the upper bound  $C$  on  $\alpha_i$  instead of  $\infty$ .

## 1.4 Non-Linear SVM

What if the surface separating the two classes is not linear? In this case we transform the data points to another high dimensional space such that the data points will become linearly separable. Let the mapping be

$$\phi: R^n \longrightarrow R^p$$

where  $p > n$ .

Using (1.11) and (1.14), in the high dimension space, we solve

$$\begin{aligned} \min L_D & \equiv \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \phi(x_i)^t \phi(x_j) - \sum_{i=1}^m \alpha_i \\ \text{subject to } & C \geq \alpha_i \geq 0 \quad \forall i = 1, \dots, m \end{aligned} \quad (1.15)$$

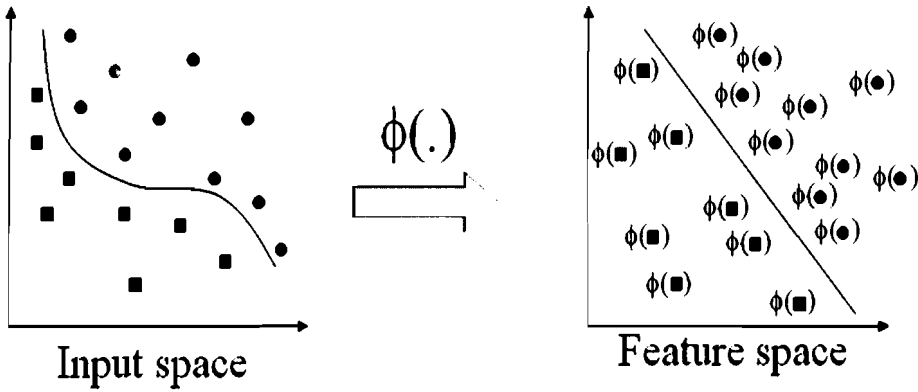


Figure 3: Mapping into a higher dimension space

Suppose in addition,  $\phi(x_i)' \phi(x_j) = k(x_i, x_j)$  where  $k(x_i, x_j)$  is a kernel function. That is the dot product in that high dimensional space is equivalent to a kernel function of the input space. This substitution allows us that we need not to know the transformation  $\phi(\cdot)$  explicitly as long as we know that the kernel function  $k(x_i, x_j)$  is equivalent to the dot product of some other high dimensional space. This is possible when the kernel function satisfies Mercer's condition (Cristianini & Taylor, 2000). There are many kernel function that can be used this way,

- Polynomial Kernel with degree  $d$  :

$$k(x, y) = (x' y + 1)^d, \quad d > 0 \text{ is a parameter}$$

- Gaussian Kernel function :

$$k(x, y) = \exp(-\mu \|x - y\|^2), \quad \mu > 0 \text{ is a parameter.}$$

Thus the dual problem (1.15) can be reformulated as

$$\min \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j k(x_i, x_j) - \sum_{i=1}^m \alpha_i$$

$$\text{subject to: } C \geq \alpha_i \geq 0 \quad \forall i = 1, \dots, m \quad (1.16)$$

with classifier function being

$$f(x) = \text{sign}((\sum_i \alpha_i y_i k(x_i, x)) + b)$$

$$\text{where } b = w' x_i - y_i (1 - \xi_i) \text{ for } i = \arg \max_j (\alpha_j)$$



The Mercer's condition can be used to determine if a function can be used as a kernel function. There exists a mapping  $\phi$  and an expansion

$$k(x, y) = \phi(x)' \phi(y),$$

if and only if, for any  $g(x)$  such that

$$\int g(x)^2 dx \text{ is finite}$$

and  $\int k(x, y)g(x)g(y)dxdy \geq 0$  must be satisfied.

## 1.5 Linear SVR

Objective of regression problems is to find out the mathematical relationship inherently lying between given input observations and their output values. With the introduction of  $\varepsilon$ -insensitive error loss function by Vapnik (Vapnik, 2000), SVM methods are successfully extended to regression problems.

Assume that a set of input samples  $\{(x_i, y_i)\}_{i=1,2,\dots,m}$  be given where for  $x_i = (x_{i1}, x_{i2}, \dots, x_{in})' \in R^n$  its observed output be  $y_i \in R$ . The linear SVR problem is the method in approximating the output by a function  $f(\cdot)$  of the form

$$f(x) = x'w + b \tag{1.17}$$

where  $w \in R^n$  and  $b \in R$  be the solution of the problem:

$$\min \frac{1}{2} w'w + C \sum_{i=1}^m \left| f(x_i) - y_i \right|_{\varepsilon} \tag{1.18}$$

where,  $|f(x_i) - y_i|_{\varepsilon} = \max\{0, |f(x_i) - y_i| - \varepsilon\}$  is the  $\varepsilon$ -insensitive error loss function and  $C > 0$  is a parameter and  $\varepsilon > 0$ .

We can write as:

$$\left| f(x_i) - y_i \right|_{\varepsilon} = |A_i w + b - y_i|_{\varepsilon}, \quad \forall i = 1, 2, \dots, m \tag{1.19}$$

where  $A \in R^{m \times n}$  is the matrix whose  $i^{\text{th}}$  row is denoted by  $A_i = x_i'$ . The equivalent constrained minimization problem of the previous unconstrained problem is:

$$\min w'w + C(e' \xi + e' \xi^*)$$

subject to:  $y - Aw - be \leq \varepsilon e + \xi,$

$$Aw + be - y \leq \varepsilon e + \xi^*, \quad (1.20)$$

$$\xi, \xi^* \geq 0 \text{ for } i = 1, 2, \dots, m$$

where  $\xi = (\xi_1, \xi_2, \dots, \xi_m)'$ ,  $\xi^* = (\xi_1^*, \xi_2^*, \dots, \xi_m^*)'$  are vectors of slack variables,  $y = (y_1, y_2, \dots, y_m)'$  is the vector of observed values and  $e = (1, \dots, 1)'$  in  $R^m$ .

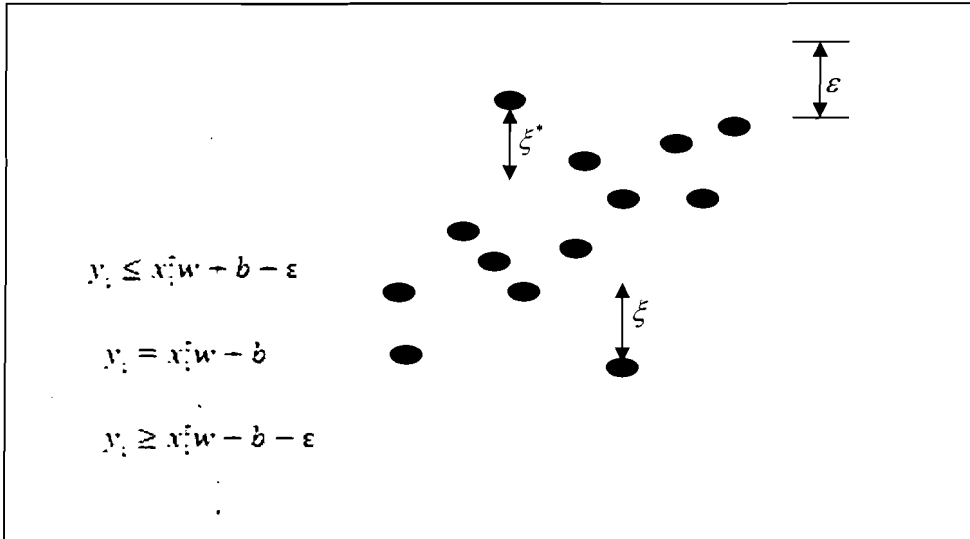


Figure 4: Graphical representation of the linear regression function and the epsilon bound.

The dual of the above problem can be formulated as

$$\min_{u_1, u_2 \in R^m} \frac{1}{2} (u_1 - u_2)' AA' (u_1 - u_2) - y' (u_1 - u_2) + \varepsilon e' (u_1 + u_2)$$

$$\text{subject to: } e' (u_1 - u_2) = 0 \text{ and } 0 \leq u_1, u_2 \leq C \quad (1.21)$$

where  $u_1, u_2 \in R^m$  are the Lagrange multipliers. Further, for any example  $x \in R^n$  its prediction is given by the following regression function

$$f(x) = x' A' (u_1 - u_2) + b \quad (1.22)$$

## 1.6 Non-Linear SVR

For the nonlinear case, the input data is mapped into a higher dimensional feature space via a kernel function  $k(.,.)$  and here in the feature space a linear SVR estimation is obtained. Let the input data be transformed into a higher dimensional feature space by the transformation  $\phi: R^n \rightarrow R^p$  where  $p > n$ . Support vector regression approximation function in the higher dimensional space is given by

$$f(x) = w' \phi(x) + b$$

Like in Linear SVR, we convert non linear SVR formulation into constrained optimization problem of the form

$$\begin{aligned} \min \quad & \frac{1}{2} w' w + C(e' \xi + e' \xi^*) \\ \text{subject to:} \quad & w' \phi(x_i) + b - y_i \leq \varepsilon + \xi_i, \\ & y_i - w' \phi(x_i) - b \leq \varepsilon + \xi_i^* \end{aligned}$$

and  $\xi_i, \xi_i^* \geq 0$  for  $i = 1, 2, \dots, m$ ,

where  $\xi = (\xi_1, \xi_2, \dots, \xi_m)'$ ,  $\xi^* = (\xi_1^*, \xi_2^*, \dots, \xi_m^*)'$  are vectors of slack variables,  $y = (y_1, y_2, \dots, y_m)'$  is the vector of observed values and  $e = (1, \dots, 1)'$  in  $R^m$ .

In this case, the SVR method can be formulated as the solution of the following dual problem:

$$\begin{aligned} \min_{(u_1, u_2) \in R^{m+m}} \quad & \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m (u_{1i} - u_{2i}) k(x_i, x_j) (u_{1j} - u_{2j}) - \sum_{i=1}^m y_i (u_{1i} - u_{2i}) + \varepsilon \sum_{i=1}^m (u_{1i} + u_{2i}) \\ \text{subject to:} \quad & e' (u_1 - u_2) = 0 \text{ and } 0 \leq u_1, u_2 \leq C \end{aligned} \quad (1.23)$$

where  $k(.,.)$  is the kernel function and  $u_1, u_2 \in R^m$  are the Lagrange multipliers. As an example of a kernel function, the Gaussian kernel defined below can be taken:

$$k(x_i, x_j) = \exp\left(-\frac{1}{2\sigma^2} \|x_i - x_j\|^2\right) \text{ for } i, j \in \{1, 2, \dots, m\} \quad (1.24)$$

where  $\sigma > 0$  is a parameter.

Using the solution of the problem, the regression estimation function  $f(.)$  for any input sample  $x \in R^n$  for the nonlinear case is obtained to be

$$f(x) = k(x', A') (u_1 - u_2) + b \quad (1.25)$$

where  $k(x', A') = (k(x, x_1), \dots, k(x, x_m))$  is a row vector in  $R^m$ .

# CHAPTER 2

## Twin Support Vector Regression

### 2.1 Linear TSVR

In this section, an efficient approach to SVR termed as Twin Support Vector Regression (TSVR) (Peng, 2010) has been introduced and it is proposed in our work to solve the problem in dual variables using Newton Method.

Consider the input set of examples  $\{(x_i, y_i)\}_{i=1, \dots, m}$  where for the example  $x_i = (x_{i1}, x_{i2}, \dots, x_{im})' \in R^n$  and its corresponding output being  $y_i \in R$ . The TSVR algorithm finds two function  $f_1(x) = x'w_1 + b_1$  and  $f_2(x) = x'w_2 + b_2$ , each one determine the  $\varepsilon$ -insensitive down-bound or up-bound regressor. Here the  $f_1(x)$  determines the  $\varepsilon_1$ -insensitive down-bound regressor, while the function  $f_2(x)$  determines the  $\varepsilon_2$ -insensitive up-bound regressor. The end regressor is decided by the mean of these two functions. A geometric interpretation is shown in Fig. 5. The constraints require the estimated function  $f_1(x)$  or  $f_2(x)$  to be at a distance of at least  $\varepsilon_1$  or  $\varepsilon_2$  from the training points. That is, the training points should be larger than the function  $f_1(x)$  at least  $\varepsilon_1$ , while they should be smaller than the function  $f_2(x)$  at least  $\varepsilon_2$ . The slack vector  $\xi_1$  or  $\xi_2$  is introduced to measure the error whenever the distance is closer than  $\varepsilon_1$  or  $\varepsilon_2$ . The second term of the objective function minimizes the sum of error variables.

The Twin SVR is proposed in ( Peng, 2010) for the linear problem which can be formulated as follows:

$$\min \frac{1}{2} (y - e\varepsilon_1 - (Aw_1 + eb_1))' (y - e\varepsilon_1 - (Aw_1 + eb_1)) + C_1 e' \xi_1$$

$$\text{subject to: } y - (Aw_1 + eb_1) \geq e\varepsilon_1 - \xi_1,$$

$$\xi_1 \geq 0 \tag{2.1}$$

$$\min \frac{1}{2} (y + e\varepsilon_2 - (Aw_2 + eb_2))' (y + e\varepsilon_2 - (Aw_2 + eb_2)) + C_2 e' \xi_2$$

$$\text{subject to: } (Aw_2 + eb_2) - y \geq e\varepsilon_2 - \xi_2,$$

$$\xi_2 \geq 0 \tag{2.2}$$

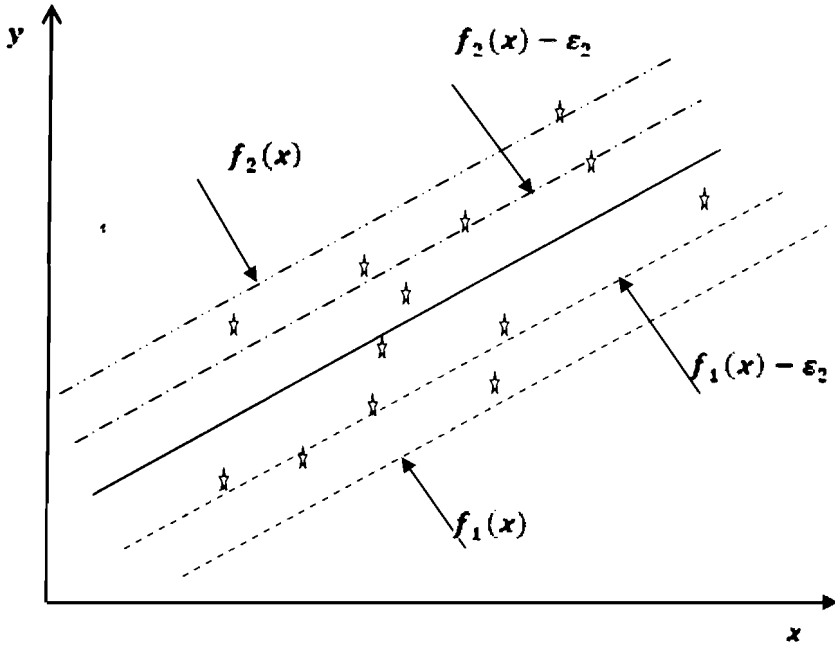


Figure 5: The geometric interpretation of TSVR

where  $\varepsilon_1, \varepsilon_2 \geq 0$  are inputs parameters;  $C_1, C_2 \geq 0$  are regularization parameters;  $y = (y_1, \dots, y_m)' \in R^m$  is the output vector and  $A \in R^{m \times n}$  is the matrix whose  $i^{th}$  row is denoted by  $A_i = x_i', \forall i = 1, \dots, m$  and  $e' = (1, \dots, 1) \in R^m$ .

To derive the dual QPPs of TSVR, introducing Lagrangian multipliers  $\alpha_i = (\alpha_{i1}, \dots, \alpha_{im})' \in R^m$  and  $\beta_i = (\beta_{i1}, \dots, \beta_{im})' \in R^m$  the Lagrangian functions corresponding to the primal problems (2.1) and (2.2) become

$$L_1(w_1, b_1, \xi_1, \alpha_1, \beta_1) = \frac{1}{2}(y - e\varepsilon_1 - (Aw_1 + eb_1))'(y - e\varepsilon_1 - (Aw_1 + eb_1)) + C_1 e' \xi_1 - \alpha_1(y - (Aw_1 + eb_1) - e\varepsilon_1 + \xi_1) - \beta_1' \xi_1 \quad (2.3)$$

$$L_2(w_2, b_2, \xi_2, \alpha_2, \beta_2) = \frac{1}{2}(y + e\varepsilon_2 - (Aw_2 + eb_2))'(y + e\varepsilon_2 - (Aw_2 + eb_2)) + C_2 e' \xi_2 - \alpha_2((Aw_2 + eb_2) - y - e\varepsilon_2 + \xi_2) - \beta_2' \xi_2 \quad (2.4)$$

satisfying the condition  $\alpha_1, \alpha_2, \beta_1, \beta_2 \geq 0$ . Applying the necessary and sufficient KKT conditions for the optimality we get

$$\begin{aligned}
-A'(y - Aw_1 - eb_1 - e\varepsilon_1) + A'\alpha_1 &= 0, \\
-e'(y - Aw_1 - eb_1 - e\varepsilon_1) + e'\alpha_1 &= 0, \\
C_1e - \alpha_1 - \beta_1 &= 0, \\
y - (Aw_1 + eb_1) &\geq e\varepsilon_1 - \xi_1, \quad \xi_1 \geq 0, \\
\alpha_1'(y - (Aw_1 + eb_1) \geq e\varepsilon_1 - \xi_1) &= 0, \quad \alpha_1 \geq 0, \\
\beta_1'\xi_1 &= 0, \quad \beta_1 \geq 0.
\end{aligned}$$

Since  $\beta_1 \geq 0$ , we have

$$0 \leq \alpha_1 \leq C_1e. \quad (2.5)$$

$$\begin{aligned}
-A'(y - Aw_2 - eb_2 + e\varepsilon_2) - A'\alpha_2 &= 0, \\
-e'(y - Aw_2 - eb_2 + e\varepsilon_2) - e'\alpha_2 &= 0, \\
C_2e - \alpha_2 - \beta_2 &= 0, \\
(Aw_2 + eb_2) - y &\geq e\varepsilon_2 - \xi_2, \quad \xi_2 \geq 0, \\
\alpha_2'((Aw_2 + eb_2) - y \geq e\varepsilon_2 - \xi_2) &= 0, \quad \alpha_2 \geq 0, \\
\beta_2'\xi_2 &= 0, \quad \beta_2 \geq 0.
\end{aligned}$$

Since  $\beta_2 \geq 0$ , we have

$$0 \leq \alpha_2 \leq C_2e. \quad (2.6)$$

Next, combining first two equation of (2.5) and (2.6) will lead to:

$$-\begin{bmatrix} A' \\ e' \end{bmatrix} \left\{ (y - e\varepsilon_1) - \begin{bmatrix} A & e \end{bmatrix} \begin{bmatrix} w_1 \\ b_1 \end{bmatrix} \right\} + \begin{bmatrix} A' \\ e' \end{bmatrix} \alpha_1 = 0 \quad (2.7)$$

$$-\begin{bmatrix} A' \\ e' \end{bmatrix} \left\{ (y + e\varepsilon_2) - \begin{bmatrix} A & e \end{bmatrix} \begin{bmatrix} w_2 \\ b_2 \end{bmatrix} \right\} - \begin{bmatrix} A' \\ e' \end{bmatrix} \alpha_2 = 0 \quad (2.8)$$

For the sake of simplicity of expression, let us define the following notation

$$\begin{aligned}
G = \begin{bmatrix} A & e \end{bmatrix}_{m \times (n+1)}, & \quad f_1 = y - e\varepsilon_1, & \quad u_1 = [w_1' \quad b_1']', \\
& \quad f_2 = y + e\varepsilon_2, & \quad u_2 = [w_2' \quad b_2']'.
\end{aligned} \quad (2.9)$$

Then we have

$$-G'f_1 + G'Gu_1 + G'\alpha_1 = 0,$$

$$\text{i.e., } u_1 = (G'G)^{-1}G'(f_1 - \alpha_1). \quad (2.10)$$

and

$$-G'f_2 + G'Gu_2 - G'\alpha_2 = 0,$$

$$\text{i.e., } u_2 = (G'G)^{-1}G'(f_2 + \alpha_2). \quad (2.11)$$

Here one can see that  $G'G$  is always positive semidefinite and it is possible that it may not be well conditioned in some situations. To overcome this ill-conditioning case, we introduce a regularization term  $\sigma I$ , where  $\sigma$  is a very small positive number, such as  $\sigma = 1e-7$ . Therefore, (2.10) and (2.11) is modified to

$$u_1 = (G'G + \sigma I)^{-1}G'(f_1 - \alpha_1), \quad (2.12)$$

$$u_2 = (G'G + \sigma I)^{-1}G'(f_2 + \alpha_2). \quad (2.13)$$

Substituting (2.12) and the above KKT conditions (2.5) into (2.3) and discarding all constant terms, the dual problem of (2.1) is presented as follows

$$\begin{aligned} \max & -\frac{1}{2}\alpha_1'G(G'G)^{-1}G'\alpha_1 + f_1'G(G'G)^{-1}G'\alpha_1 - f_1'\alpha_1 \\ \text{subject to: } & 0 \leq \alpha_1 \leq eC_1 \end{aligned} \quad (2.14)$$

Similarly, substituting (2.13) and the above KKT conditions (2.6) into (2.4) and discarding all constant terms, the dual problem of (2.2) can be written as

$$\begin{aligned} \max & -\frac{1}{2}\alpha_2'G(G'G)^{-1}G'\alpha_2 - f_2'G(G'G)^{-1}G'\alpha_2 + f_2'\alpha_2 \\ \text{subject to: } & 0 \leq \alpha_2 \leq eC_2 \end{aligned} \quad (2.15)$$

Solving the equation (2.14) and (2.15) for  $\alpha_1$  and  $\alpha_2$ , the coefficients of down- and up-

bounds, i.e.,  $\begin{bmatrix} w_1 \\ b_1 \end{bmatrix}$  and  $\begin{bmatrix} w_2 \\ b_2 \end{bmatrix}$ , can be obtained by substituting the results into the following

formula

$$\begin{aligned} \begin{bmatrix} w_1 \\ b_1 \end{bmatrix} &= (G'G)^{-1}G'(f_1 - \alpha_1), \\ \begin{bmatrix} w_2 \\ b_2 \end{bmatrix} &= (G'G)^{-1}G'(f_2 + \alpha_2). \end{aligned}$$

Thus, the training procedure of TSVR has completed and for a new data sample, TSVR use the average of its down- and up-bounds to evaluate its value, which is presented by

$$f_1(x) = x'w_1 + b_1 = [x' \quad 1] \begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = [x' \quad 1](G'G)^{-1}G'(f_1 - \alpha_1) \quad (2.16)$$

$$f_2(x) = x'w_2 + b_2 = [x' \quad 1] \begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = [x' \quad 1](G'G)^{-1}G'(f_2 + \alpha_2) \quad (2.17)$$

Finally, the regression estimation function is defined to be:

$$f(x) = \frac{1}{2}(f_1(x) + f_2(x)) \text{ for any } x \in R^n.$$

## 2.2 Kernel TSVR

As it was stated earlier, the extension of the above study to nonlinear regressor is possible by mapping the input data into a higher dimensional feature space using a kernel function  $k(\dots)$  and applying the linear TSVR in the feature space. Any function satisfying Mercer's condition (Vapnik, 2000) can be used as a kernel function. In our analysis, the Gaussian kernel function defined by

$$k(x, y) = \exp\left(-\frac{1}{2\sigma^2}\|x - y\|^2\right) \text{ for } x, y \in R^n$$

is used, where  $\sigma > 0$  is a parameter.

In order to extend our results to nonlinear regressors, following the approach of Mangasarian & Musicant (2001) the nonlinear regression estimation functions are obtained by:

$$f_1(x) = k(x', A')w_1 + b_1 \quad \text{and} \quad f_2(x) = k(x', A')w_2 + b_2$$

where  $k(x', A') = (k(x, x_1), \dots, k(x, x_m))$  is a row vector and  $k(\dots)$  is a kernel function.

The twin SVR for the nonlinear case will be formulated as a pair of minimization problems:

$$\begin{aligned} & \min \frac{1}{2}(y - e\varepsilon_1 - (k(A, A')w_1 + eb_1))(y - e\varepsilon_1 - (k(A, A')w_1 + eb_1)) + C_1 e' \xi_1 \\ & \text{subject to:} \quad y - (k(A, A')w_1 + eb_1) \geq e\varepsilon_1 - \xi_1, \\ & \quad \quad \quad \xi_1 \geq 0 \end{aligned} \quad (2.18)$$

$$\begin{aligned} & \min \frac{1}{2}(y + e\varepsilon_2 - (k(A, A')w_2 + eb_2))(y - e\varepsilon_2 - (k(A, A')w_2 + eb_2)) + C_2 e' \xi_2 \\ & \text{subject to:} \quad (k(A, A')w_2 + eb_2) - y \geq e\varepsilon_2 - \xi_2, \\ & \quad \quad \quad \xi_2 \geq 0 \end{aligned} \quad (2.19)$$



where  $k(A, A')$  is the kernel matrix whose  $(i, j)^{th}$  component  $k(A, A')_{ij} = k(x_i, x_j)$ .

Proceeding as in the linear case and replacing  $[A \ e]_{m \times (n+1)}$  by  $[K(A, A') \ e]_{m \times (n+1)}$  the dual problem for the nonlinear case can be written as a pair of minimization problems of the form

$$\min_{0 \leq u_1 \in R^m} L_1(u_1) = \frac{1}{2} u_1' Q_1 u_1 - r_1' u_1 \quad (2.20)$$

and

$$\min_{0 \leq u_2 \in R^m} L_2(u_2) = \frac{1}{2} u_2' Q_2 u_2 - r_2' u_2 \quad (2.21)$$

where

$$Q_1 = \frac{I}{C_1} + G(G'G)^{-1}G', \quad r_1 = (G(G'G)^{-1}G' - I)(y - e\varepsilon_1),$$

$$Q_2 = \frac{I}{C_2} + G(G'G)^{-1}G', \quad r_2 = (I - G(G'G)^{-1}G')(y - e\varepsilon_2)$$

and  $G = [K(A, A') \ e]_{m \times (m+1)}$  is an augmented matrix.

Further in this case, the primal variables can be expressed as

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = (G'G)^{-1}G'(y - e\varepsilon_1 - u_1) \quad (2.22)$$

and

$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = (G'G)^{-1}G'(y + \varepsilon_2 e + u_2). \quad (2.23)$$

Now using the equation (2.16) & (2.17), we get

$$f_1(x) = k(x', A')w_1 + b_1 = \begin{bmatrix} k(x', A') & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ b_1 \end{bmatrix}$$

$$\text{or} \quad f_1(x) = \begin{bmatrix} k(x', A') & 1 \end{bmatrix} (G'G)^{-1}G'(y - e\varepsilon_1 - u_1) \quad (2.24)$$

$$\text{and} \quad f_2(x) = k(x', A')w_2 + b_2 = \begin{bmatrix} k(x', A') & 1 \end{bmatrix} \begin{bmatrix} w_2 \\ b_2 \end{bmatrix}$$

$$\text{or} \quad f_2(x) = \begin{bmatrix} k(x', A') & 1 \end{bmatrix} (G'G)^{-1}G'(y + e\varepsilon_2 + u_2) \quad (2.25)$$

In this case, the regression estimation function is defined to be:

$$f(x) = \frac{1}{2}(f_1(x) + f_2(x)) \text{ for any } x \in R^n.$$

# CHAPTER 3

## Implicit Lagrangian Twin Support Vector Regression (LTSVR) By Newton Method

### 3.1 Linear LTSVR

Twin support vector regression (TSVR) (Peng, 2010) was proposed as a novel regressor that tries to find a pair of nonparallel planes, i.e.,  $\varepsilon$ -insensitive up- and down-bounds, by solving two related SVM-type problems. It results in a pair of small sized QPPs rather than a single large one as in classical SVR. It is inspired by TWSVM (Jayadeva et al., 2007) wherein a pair of nonparallel functions corresponding to the  $\varepsilon$ -insensitive down- and up- bounds is determined. The TSVR formulation leads to the solution of a pair of dual QPPs of smaller size rather than a single large one as in the standard SVR. This strategy makes TSVR works faster than SVR with the added advantage of better generalization ability over SVR (Peng, 2010). By making a slight change in the formulation of TSVR to become a strongly convex optimization problem and employing smooth technique, a new formulation called smooth TSVR (STSVR) has been proposed as an unconstrained minimization problem in (Chen et al., 2011) and its solution is obtained by the well-known Newton-Armijo algorithm. For the work on the formulation of TSVR as a pair of linear programming problems, we refer the reader to (Zhong et al., 2011).

Motivated by the study on implicit Lagrangian SVM for classification problems suggested by Fung and Mangasarian (2003) in this work we proposed its extension on TSVR by formulating a pair of unconstrained minimization problems in dual variables whose solutions will be obtained by Newton method.

The Twin SVR is proposed in ( Peng, 2010) for the linear problem in 1-norm which can be formulated as follows:

$$\begin{aligned} \min & \frac{1}{2}(y - e\varepsilon_1 - (Aw_1 + eb_1))(y - e\varepsilon_1 - (Aw_1 + eb_1)) + C_1 e' \xi_1 \\ \text{subject to:} & \quad y - (Aw_1 + eb_1) \geq e\varepsilon_1 - \xi_1, \\ & \quad \xi_1 \geq 0 \end{aligned} \tag{3.1}$$

$$\min \frac{1}{2} (y + e\varepsilon_2 - (Aw_2 + eb_2))' (y + e\varepsilon_2 - (Aw_2 + eb_2)) + C_2 e' \xi_2$$

$$\text{subject to: } (Aw_2 + eb_2) - y \geq e\varepsilon_2 - \xi_2,$$

$$\xi_2 \geq 0 \quad (3.2)$$

where  $\varepsilon_1, \varepsilon_2 \geq 0$  are inputs parameters;  $C_1, C_2 \geq 0$  are regularization parameters;

$y = (y_1, \dots, y_m)' \in R^m$  is the output vector and  $A \in R^{m \times n}$  is the matrix whose  $i^{\text{th}}$  row is denoted by  $A_i = x_i', \forall i = 1, \dots, m$  and  $e' = (1, \dots, 1) \in R^m$ .

Following the approach of (Balasundaram & Kapil, 2010; Mangasarian & Musicant, 2001; Musicant & Feinberg, 2004) where 2-norm is considered instead of 1-norm, we propose to reformulate TSVR of the following form:

$$\min \frac{1}{2} (y - e\varepsilon_1 - (Aw_1 + eb_1))' (y - e\varepsilon_1 - (Aw_1 + eb_1)) + \frac{C_1}{2} \xi_1' \xi_1$$

$$\text{subject to: } y - (Aw_1 + eb_1) \geq e\varepsilon_1 - \xi_1 \quad (3.3)$$

and

$$\min \frac{1}{2} (y + e\varepsilon_2 - (Aw_2 + eb_2))' (y + e\varepsilon_2 - (Aw_2 + eb_2)) + \frac{C_2}{2} \xi_2' \xi_2$$

$$\text{subject to: } (Aw_2 + eb_2) - y \geq e\varepsilon_2 - \xi_2. \quad (3.4)$$

By introducing the Lagrange multipliers  $u_1 = (u_{11}, \dots, u_{1m})' \in R^m$  and

$u_2 = (u_{21}, \dots, u_{2m})' \in R^m$  consider the Lagrangian functions:

$$\begin{aligned} L_1(w_1, b_1, \xi_1) = & \frac{1}{2} (y - e\varepsilon_1)' (y - e\varepsilon_1) - (y - e\varepsilon_1)' (Aw_1 + eb_1) - (Aw_1 + eb_1)' (y - e\varepsilon_1) + \\ & \frac{1}{2} (Aw_1 + eb_1)' (Aw_1 + eb_1) + \frac{C_1}{2} \xi_1' \xi_1 - u_1' [y - (Aw_1 + eb_1) - e\varepsilon_1 + \xi_1] \end{aligned} \quad (3.5)$$

$$\begin{aligned} L_2(w_2, b_2, \xi_2) = & \frac{1}{2} (y + e\varepsilon_2)' (y + e\varepsilon_2) - (y + e\varepsilon_2)' (Aw_2 + eb_2) + \frac{1}{2} (Aw_2 + eb_2)' (Aw_2 + eb_2) + \\ & \frac{C_2}{2} \xi_2' \xi_2 - u_2' [(Aw_2 + eb_2) - y - e\varepsilon_2 + \xi_2] \end{aligned} \quad (3.6)$$

Now using the property that the partial derivatives of  $L_1$  and  $L_2$  with respect to primal variables will be zero as the optimality condition i.e

$$\frac{\partial L_1}{\partial w_1} = 0 \Rightarrow A' u_1 = A' \left\{ (y - e\varepsilon_1) - \begin{bmatrix} A & e \end{bmatrix} \begin{bmatrix} w_1 \\ b_1 \end{bmatrix} \right\} \quad (3.7)$$

$$\frac{\partial L_1}{\partial b_1} = 0 \Rightarrow e' u_1 = e' \left\{ (y - e \varepsilon_1) - [A \quad e] \begin{bmatrix} w_1 \\ b_1 \end{bmatrix} \right\}. \quad (3.8)$$

$$(3.7) \& (3.8) \Rightarrow \begin{bmatrix} A' \\ e' \end{bmatrix} u_1 = \begin{bmatrix} A' \\ e' \end{bmatrix} \left\{ (y - e \varepsilon_1) - [A \quad e] \begin{bmatrix} w_1 \\ b_1 \end{bmatrix} \right\}$$

$$\text{i.e.} \quad \begin{bmatrix} A' \\ e' \end{bmatrix} [A \quad e] \begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = \begin{bmatrix} A' \\ e' \end{bmatrix} \{ (y - e \varepsilon_1 - u_1) \} \quad (3.9)$$

Define the augmented matrix  $G = [A \quad e]_{m \times (n+1)}$ . Then the equation (3.9) will become

$$G' G \begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = G' (y - e \varepsilon_1 - u_1) \quad (3.10)$$

Assume  $(G' G)^{-1}$  exists. Then we have

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = (G' G)^{-1} G' (y - e \varepsilon_1 - u_1) \quad (3.11)$$

$$\text{Finally} \quad \frac{\partial L_1}{\partial \xi_1} = 0 \Rightarrow \xi_1 = \frac{u_1}{c_1} \quad (3.12)$$

Similarly for  $L_2$

$$\frac{\partial L_2}{\partial w_2} = 0 \Rightarrow A' [A \quad e] \begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = A' (y + \varepsilon_2 e + u_2) \quad (3.13)$$

$$\frac{\partial L_2}{\partial b_2} = 0 \Rightarrow e' [A \quad e] \begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = e' (y + \varepsilon_2 e + u_2) \quad (3.14)$$

$$(3.13) \& (3.14) \Rightarrow \begin{bmatrix} A' \\ e' \end{bmatrix} [A \quad e] \begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = \begin{bmatrix} A' \\ e' \end{bmatrix} (y + \varepsilon_2 e + u_2) \quad (3.15)$$

Then the equation (3.15) will become

$$G' G \begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = G' (y + \varepsilon_2 e + u_2) \quad (3.16)$$

Assume  $(G' G)^{-1}$  exists. Then we have

$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = (G' G)^{-1} G' (y + \varepsilon_2 e + u_2) \quad (3.17)$$

$$\text{Finally} \quad \frac{\partial L_2}{\partial \xi_2} = 0 \Rightarrow \xi_2 = \frac{u_2}{c_2} \quad (3.18)$$

So using the equation (3.10), (3.11), (3.17), (3.18) in (3.5) & (3.6), finally we get

$$\begin{aligned} \max_{u_1 \geq 0, u_1 \in R^m} L_1(u_1) &= \frac{1}{2} \left[ (y - e\varepsilon_1)'(y - e\varepsilon_1) - (y - e\varepsilon_1)' G(G'G)^{-1} G'(y - e\varepsilon_1) \right] \\ &\quad + (y - \varepsilon_1 e)' G(G'G)^{-1} G' u_1 - (y - \varepsilon_1 e)' u_1 - \frac{1}{2} u_1' \left( \frac{I}{C_1} + G(G'G)^{-1} G' \right) u_1 \end{aligned} \quad (3.19)$$

$$\begin{aligned} \max_{u_2 \geq 0, u_2 \in R^m} L_2(u_2) &= \frac{1}{2} \left[ (y - e\varepsilon_2)'(y - e\varepsilon_2) - (y - e\varepsilon_2)' G(G'G)^{-1} G'(y + \varepsilon_2 e) \right] \\ &\quad - (y + \varepsilon_2 e)' G(G'G)^{-1} G' u_2 + (y + \varepsilon_2 e)' u_2 - \frac{1}{2} u_2' \left( \frac{I}{C_2} + G(G'G)^{-1} G' \right) u_2. \end{aligned} \quad (3.20)$$

$\max_{u_1 \geq 0, u_1 \in R^m} L_1(u_1)$  and  $\max_{u_2 \geq 0, u_2 \in R^m} L_2(u_2)$  will be same as the following minimization problem :

$$\min_{u_1 \geq 0} \frac{1}{2} u_1' \left( \frac{I}{C_1} + G(G'G)^{-1} G' \right) u_1 - \left\{ (y - e\varepsilon_1)' G(G'G)^{-1} G' - (y - e\varepsilon_1)' \right\} u_1 \quad (3.21)$$

$$\min_{u_2 \geq 0} \frac{1}{2} u_2' \left( \frac{I}{C_2} + G(G'G)^{-1} G' \right) u_2 - \left\{ -(y + e\varepsilon_1)' G(G'G)^{-1} G' + (y + e\varepsilon_1)' \right\} u_2 \quad (3.22)$$

which can be written in the following form:

$$\begin{aligned} \min_{u_1 \geq 0} \frac{1}{2} u_1' Q_1 u_1 - r_1' u_1, \\ \min_{u_2 \geq 0} \frac{1}{2} u_2' Q_2 u_2 - r_2' u_2 \end{aligned} \quad (3.23)$$

where

$$\begin{aligned} Q_1 &= \frac{I}{C_1} + G(G'G)^{-1} G', \\ r_1 &= \left( G(G'G)^{-1} G' - I \right) (y - e\varepsilon_1), \\ Q_2 &= \frac{I}{C_2} + G(G'G)^{-1} G', \\ r_2 &= \left( I - G(G'G)^{-1} G' \right) (y - e\varepsilon_2) \end{aligned} \quad (3.24)$$

and  $G = [A \ e]$  is an augmented matrix.

Further in this case we have

$$f_1(x) = x'w_1 + b_1 = [x' \quad 1] \begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = [x' \quad 1] (G'G)^{-1} G' (y - e\varepsilon_1 - u_1) \quad (3.25)$$

$$f_2(x) = x'w_2 + b_2 = [x' \quad 1] \begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = [x' \quad 1] (G'G)^{-1} G' (y + e\varepsilon_2 + u_2) \quad (3.26)$$

Finally, the regression estimation function is defined to be:

$$f(x) = \frac{1}{2} (f_1(x) + f_2(x)) \text{ for any } x \in R^n.$$

### 3.2. Non-Linear LTSVR

In order to extend our results to nonlinear regressors, we consider the following kernel-generated functions :

$$f_1(x) = k(x', A')w_1 + b_1 \text{ and } f_2(x) = k(x', A')w_2 + b_2,$$

where  $k(x', A') = (k(x, x_1), \dots, k(x, x_m))$  is a row vector and  $k(\dots)$  is a kernel function.

The primal TSVR in 2-norm is comprised of the following pair of constrained minimization problems:

$$\min_{(w_1, b_1, \xi_1) \in R^{n+1+m}} \frac{1}{2} (y - e\varepsilon_1 - (k(A, A')w_1 + eb_1))^2 + \frac{C_1}{2} \xi_1' \xi_1$$

subject to:  $y - (k(A, A')w_1 + eb_1) \geq e\varepsilon_1 - \xi_1,$  (3.27)

and

$$\min_{(w_2, b_2, \xi_2) \in R^{n+1+m}} \frac{1}{2} (y + e\varepsilon_2 - (k(A, A')w_2 + eb_2))^2 + \frac{C_2}{2} \xi_2' \xi_2$$

subject to:  $(k(A, A')w_2 + eb_2) - y \geq e\varepsilon_2 - \xi_2$  (3.28)

where  $k(A, A')$  is the kernel matrix whose  $(i, j)^{th}$  component  $k(A, A')_{ij} = k(x_i, x_j)$ .

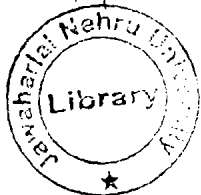
By introducing the Lagrange multipliers  $u_1 = (u_{11}, \dots, u_{1m})' \in R^m$  and

$u_2 = (u_{21}, \dots, u_{2m})' \in R^m$  consider the Lagrangian functions:

$$L_1(w_1, b_1, \xi_1) = \frac{1}{2} (y - e\varepsilon_1)' (y - e\varepsilon_1) - (y - e\varepsilon_1)' (k(A, A')w_1 + eb_1) - (k(A, A')w_1 + eb_1)' (y - e\varepsilon_1) + \frac{1}{2} (k(A, A')w_1 + eb_1)' (k(A, A')w_1 + eb_1) + \frac{C_1}{2} \xi_1' \xi_1 - u_1' [y - (k(A, A')w_1 + eb_1) - e\varepsilon_1 + \xi_1]$$

(3.29)

TH-20618



THESES/DISSERTATION  
TH20618

$$L_2(w_2, b_2, \xi_2) = \frac{1}{2}(y + e\varepsilon_2)'(y + e\varepsilon_2) - (y + e\varepsilon_2)'(k(A, A')w_2 + eb_2) + \frac{1}{2}(k(A, A')w_2 + eb_2)'(k(A, A')w_2 + eb_2) + \frac{C_2}{2}\xi_2'\xi_2 - u_2'[(k(A, A')w_2 + eb_2) - y - e\varepsilon_2 + \xi_2] \quad (3.30)$$

Following the same procedure as of the linear case and using the property that the partial derivatives of  $L_1$  and  $L_2$  with respect to primal variables will be zero we get,

$$\max_{u_1 \geq 0, u_1 \in R^m} L_1(u_1) = \frac{1}{2}[(y - e\varepsilon_1)'(y - e\varepsilon_1) - (y - e\varepsilon_1)'G(G'G)^{-1}G'(y - \varepsilon_1 e)] + (y - \varepsilon_1 e)'G(G'G)^{-1}G'u_1 - (y - \varepsilon_1 e)'u_1 - \frac{1}{2}u_1' \left( \frac{I}{C_1} + G(G'G)^{-1}G' \right) u_1 \quad (3.31)$$

$$\max_{u_2 \geq 0, u_2 \in R^m} L_2(u_2) = \frac{1}{2}[(y - e\varepsilon_2)'(y - e\varepsilon_2) - (y - e\varepsilon_2)'G(G'G)^{-1}G'(y + \varepsilon_2 e)] - (y + \varepsilon_2 e)'G(G'G)^{-1}G'u_2 + (y + \varepsilon_2 e)'u_2 - \frac{1}{2}u_2' \left( \frac{I}{C_2} + G(G'G)^{-1}G' \right) u_2 \quad (3.32)$$

which will be equivalent to the following minimization problem

$$\min_{u_1 \geq 0} \frac{1}{2}u_1' \left( \frac{I}{C_1} + G(G'G)^{-1}G' \right) u_1 - \left\{ (y - e\varepsilon_1)'G(G'G)^{-1}G' - (y - e\varepsilon_1)' \right\} u_1 \quad (3.33)$$

$$\min_{u_2 \geq 0} \frac{1}{2}u_2' \left( \frac{I}{C_2} + G(G'G)^{-1}G' \right) u_2 - \left\{ (y + e\varepsilon_1)'G(G'G)^{-1}G' + (y + e\varepsilon_1)' \right\} u_2 \quad (3.34)$$

This can be written in the following simpler form:

$$\begin{aligned} \min_{u_1 \geq 0} \frac{1}{2}u_1'Q_1u_1 - r_1'u_1 \\ \min_{u_2 \geq 0} \frac{1}{2}u_2'Q_2u_2 - r_2'u_2 \end{aligned} \quad (3.35)$$

where

$$\begin{aligned} Q_1 &= \frac{I}{C_1} + G(G'G)^{-1}G', \\ r_1 &= (G(G'G)^{-1}G' - I)(y - e\varepsilon_1), \\ Q_2 &= \frac{I}{C_2} + G(G'G)^{-1}G', \\ r_2 &= (I - G(G'G)^{-1}G')(y - e\varepsilon_2) \end{aligned} \quad (3.36)$$

and  $G = [K(A, A') \quad e]_{m \times (m+1)}$  is an augmented matrix.

The primal variables can be expressed as :

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = (G'G)^{-1} G'(y - e\varepsilon_1 - u_1)$$

and

$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = (G'G)^{-1} G'(y + \varepsilon_2 e + u_2)$$

Further in this case we have

$$f_1(x) = k(x', A')w_1 + b_1 = [k(x', A') \quad 1] \begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = [k(x', A') \quad 1] (G'G)^{-1} G'(y - e\varepsilon_1 - u_1) \quad (3.37)$$

$$f_2(x) = k(x', A')w_2 + b_2 = [k(x', A') \quad 1] \begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = [k(x', A') \quad 1] (G'G)^{-1} G'(y + e\varepsilon_2 + u_2) \quad (3.38)$$

Finally, the regression estimation function is defined to be:

$$f(x) = \frac{1}{2}(f_1(x) + f_2(x)) \text{ for any } x \in R^n.$$

Applying the Karush-Kuhn-Tucker (KKT) necessary and sufficient optimal condition for the dual TSVM will lead to the following pair of classical complementarity problems (Mangasarian, 1994):

$$0 \leq u_1 \perp (Q_1 u_1 - r_1) \geq 0 \text{ and } 0 \leq u_2 \perp (Q_2 u_2 - r_2) \geq 0. \quad (3.39)$$

However, using the well-known identity between two vectors  $u, v$ :

$$0 \leq u \perp v \geq 0 \text{ if and only if } u = (u - \alpha v)_+ \text{ for any } \alpha > 0,$$

the solution of the following equivalent pair of problems will be considered (Mangasarian & Musicant, 2001): for any  $\alpha_1, \alpha_2 > 0$ ,

$$Q_1 u_1 - r_1 = ((Q_1 u_1 - r_1) - \alpha_1 u_1)_+ \text{ and } Q_2 u_2 - r_2 = ((Q_2 u_2 - r_2) - \alpha_2 u_2)_+. \quad (3.40)$$

It turns out that (3.40) become necessary and sufficient conditions to be satisfied by the unconstrained minimum of the following pair of implicit Lagrangian (Mangasarian & Solodov, 1993) associated to the pair of dual problems (3.35): for any  $\alpha_1, \alpha_2 > 0$ ,



$$\min_{u \in R^m} L_1(u) = \frac{1}{2} u' Q_1 u - r_1' u + \frac{1}{2\alpha_1} (\| ((Q_1 - \alpha_1 I)u - r_1)_+ \|^2 - \| Q_1 u - r_1 \|^2) \quad (3.41)$$

and

$$\min_{u \in R^m} L_2(u) = \frac{1}{2} u' Q_2 u - r_2' u + \frac{1}{2\alpha_2} (\| ((Q_2 - \alpha_2 I)u - r_2)_+ \|^2 - \| Q_2 u - r_2 \|^2). \quad (3.42)$$

### 3.3. Newton Method

In this section we solve the pair of dual QPPs defined by (3.41) and (3.42) using the well-known Newton method (Fung & Mangasarian, 2003).

For  $k = 1, 2$ , the basic Newton's step of the iterative algorithm is in determining the unknown  $u^{i+1}$  at the  $(i+1)^{th}$  iterative using the current  $i^{th}$  iterative  $u^i$  using

$$\nabla L_k(u^i) + \nabla^2 L_k(u^i)(u^{i+1} - u^i) = 0 \quad \forall i = 0, 1, 2, \dots$$

Now, for  $u \in R^m$  one can obtain the gradient of  $L_k(u)$  as :

$$\nabla L_k(u) = \frac{(\alpha_k I - Q_k)}{\alpha_k} [(Q_k u - r_k) - (Q_k u - \alpha_k u - r_k)_+].$$

Note that for  $k = 1, 2$ , the gradient  $\nabla L_k(u)$  is not differentiable and therefore the Hessian matrix of second order partial derivatives of  $L_k(u)$  is not defined in the usual sense. However, a generalized Hessian of  $L_k(u)$  in the sense of (Hiriart-Urruty et al., 1984) exists and is given by: for all  $u \in R^m$ ,

$$\nabla^2 L_k(u) = \frac{(\alpha_k I - Q_k)}{\alpha_k} (Q_k + \text{diag}((Q_k - \alpha_k I)u - r_k)_+ (\alpha_k I - Q_k)),$$

where  $\text{diag}(\cdot)$  is a diagonal matrix of order  $m$ .

Using the property that the matrix  $Q_k \in R^{m \times m}$  defined by (3.36) is positive-definite and choosing the parameter  $\alpha_k$  satisfying the condition (Fung & Mangasarian, 2003):  $\alpha_k > \|Q_k\|$  where  $k = 1, 2$ , the Newton's iterative step can be rewritten in the following simpler form: for  $i = 0, 1, 2, \dots$

$$u^{i+1} = u^i - \partial h_k(u^i)^{-1} h_k(u^i), \quad (3.43)$$

wherein for any vector  $u \in R^m$ ,

$$h_k(u) = (Q_k u - r_k) - (Q_k u - \alpha_k u - r_k)_+$$

is a vector in  $R^m$  and  $\partial h_k(u)$  is a matrix of order  $m$  defined by:

$$\partial h_k(u) = (Q_k + \text{diag}((Q_k - \alpha_k I)u - r_k), (\alpha_k I - Q_k)).$$

By defining the following diagonal matrices of order  $m$ :

$$D_k = \text{diag}(Q_k u - \alpha_k u - r_k), F_k = \alpha_k D_k + \left( \frac{I - D_k}{C_k} \right) \text{ and } E_k = F_k^{-1}(I - D_k)$$

where  $k = 1, 2$ , we can determine

$$\partial h_k(u)^{-1} = (I + E_k H)^{-1} F_k^{-1}.$$

Now, we state the Newton iterative Algorithm with Armijo stepsize (Balasundaram & Rampal, 2010; Fung & Mangasarian, 2003; Lee et al., 2005) for solving each of the pair of unconstrained minimization problems defined by (3.41) & (3.42).

**Newton Algorithm with Armijo stepsize** for solving (3.41) & (3.42) with  $k = 1, 2$ .

Choose any initial guess  $u^0 \in R^m$

(i) Stop the iteration if  $h_k(u^i) = 0$

Else

Determine the direction vector  $d^i \in R^m$  as the solution of the following linear system of equation in  $m$  variables

$$\partial h_k(u^i) d^i = -h_k(u^i)$$

(ii) Armijo stepsize: Define

$$u^{i+1} = u^i + \lambda_i d^i$$

where  $\lambda_i = \max\left\{1, \frac{1}{2}, \frac{1}{4}, \dots\right\}$  is the stepsize in which

$$L_k(u^i) - L_k(u^i + \lambda_i d^i) \geq -\delta \lambda_i \nabla L_k(u^i) d^i$$

$$\text{and } \delta \in \left(0, \frac{1}{2}\right).$$

With the assumption that: for  $k = 1, 2$ ,  $\alpha_k > \|Q_k\|$ , the finite termination of the above Newton algorithm with Armijo step size will follow as a simple extension of the result of (Fung & Mangasarian, 2003).

Note that the matrix  $G'G$  is positive semi-definite, It is possible that in certain applications the matrix may be ill-conditioned and therefore its inverse may not exist. In such cases, however, by introducing regularization term  $\delta I$  the modified matrix  $(\delta I + G'G)$  will be used where  $\delta$  is chosen to be a very small positive number.

# CHAPTER 4

## Experimental Results

### 4.1 Introduction

To demonstrate the performance of the proposed method we compare it with TSVR and the SVR on several synthetic and real-world benchmark data sets. In Table 1, the list of functions considered for generating synthetic datasets were summarized. Also we carried out experiments on several real-world datasets- Box Jenkins gas furnace data; Google, IBM, Intel, RedHat, MS stock and S&P 500 financial time series data; Sunspot and SantaFeA time series forecasting data; the data generated by the Mackey glass and Lorenz differential equations; and Machine CPU, Bodyfat, Concrete compression strength (CS), Abalone and Kin-fh data.

All the regression algorithms are implemented in MATLAB R2009a environment on Windows XP OS with 2.8 GHz Intel P4 processor having 1.99 GB RAM. Although Newton-Armijo algorithm converges globally, all the experiments were performed without Armijo stepsize. TSVR was implemented using optimization tool box of MATLAB and for the implementation of standard SVR, however, LIBSVM (Chang and Lin, 2001) was used. For determining the optimal values of the regularization and kernel parameters, ten fold crossvalidation procedure is adopted in all our experiments. Gaussian kernel  $k(x, y) = \exp(-\frac{1}{2\sigma^2} \|x - y\|^2)$  is used to construct nonlinear regressor. The value of  $\varepsilon$  was taken to be 0.01. The 2-norm root mean square error (RMSE) was selected as the measure of prediction performance and it was calculated using the following formula

$$RMSE = \sqrt{\frac{1}{P} \sum_{i=1}^P (y_i - \tilde{y}_i)^2},$$

where  $P$  is the number of test samples and,  $y_i$  and  $\tilde{y}_i$  are the observed and their corresponding predicted values respectively. Since more parameters are to be selected in the proposed implicit LTSVR and TSVR in comparison to SVR, which will result in slower model selection speed, in all experiments we set  $\varepsilon_1 = \varepsilon_2 = 0.01$  and  $C_1 = C_2$ . For SVR, we assumed  $\varepsilon = 0.01$ . The optimal parameter values were determined by performing 10-fold cross validation methodology. In fact, for each dataset, the best

prediction performance on the training set was obtained by varying the regularization parameter  $C_1 = C_2 = C$  and kernel parameter  $\sigma$  from the sets  $\{10^{-3}, 10^{-2}, \dots, 10^3\}$  and  $\{2^{-10}, 2^{-9}, \dots, 2^{10}\}$  respectively and by choosing these optimal parameter values for training, the RMSE on the test set was calculated.

## 4.2. Experiments on synthetic data sets

In order to investigate the performance of the proposed implicit LTSVR on synthetic datasets, tabulated in Table 1, first we consider the following function for approximation (Li et al, 2011):

$$y(x) = \frac{4}{|x|+2} + \cos(2x) + \sin(3x), \text{ for } x \in [-10, 10].$$

In the experiments, 1200 samples are generated according to the functions listed in Table 1. In which we use 200 samples for the training and 1000 samples for testing. The constraints on the attributes are also shown in the Table 1. Then, the observed values for the training data were polluted by adding two different kinds of noises namely: uniform noise over the interval  $[-0.2, 0.2]$  and Gaussian noise with mean zero and standard deviation 0.2. Test data was assumed to be noise-free. The approximation functions obtained by the standard SVR, TSVR and the implicit LTSVR for uniform and Gaussian additive noises are illustrated in Fig.6a and Fig.6b respectively along with the exact function. The noisy training examples are indicated by the symbol ‘o’. The optimal parameter values and the final results obtained on the set were summarized in Table 2.

To further analyze the performance of implicit LTSVR, another 5 synthetic datasets of 1200 samples each were generated using functions whose definitions are given in Table 1. Like in previous case, 200 samples were used for training and 1000 samples for testing. The observed value  $y_i$  for the training example  $x_i$  was obtained as follows:

$$y_i = f(x_i) + \xi_i, \quad \xi_i \sim N(0, 0.2^2)$$

$$y_i = f(x_i) + \xi_i, \quad \xi_i \sim U[-0.2, 0.2]$$

where  $U[a, b]$  means the uniform distribution over the interval spanned by a and b and  $N(\mu, \sigma^2)$  represents a Normal distribution with means  $\mu$  and variance  $\sigma^2$ . By the tuning procedure explained above, the optimal parameter values were determined and using these values the RMSE on the test set was calculated. The results were summarized in Table 2.

For function 2 defined by

$$y(x) = \frac{4}{|x| + 2} + \cos(2x), \quad \text{for } x \in [-10, 10].$$

the approximation functions obtained by the standard SVR, TSVR and the implicit LTSVR for uniform and Gaussian additive noises are illustrated in Fig. 7a and Fig. 7b respectively.

Name	Function Definition	Domain of Definition
Function 1	$\frac{4}{ x +2} + \cos(2x) + \sin(3x)$	$x \in [-10,10]$
Function 2	$\frac{4}{ x +2} + \cos(2x)$	$x \in [-10,10]$
Function 3	$0.79 + 1.27x_1x_2 + 1.56x_1x_4 + 3.42x_2x_5 + 2.06x_3x_4x_5$	$x_i \in [-2,2], i \in \{1,2,3,4,5\}$
Function 4	$\frac{1 + \sin(2x_1 + 3x_2)}{3.5 + \sin(x_1 - x_2)}$	$x_1, x_2 \in [-2,2]$
Function 5	$1.3356(e^{3(x_2-0.5)} \sin(4\pi(x_2 - 0.9)^2) + 1.5(1 - x_1) + e^{(2x_1-1)} \sin(3\pi(x_1 - 0.6)^2))$	$x_1, x_2 \in [-0.5,0.5]$
Function 6	$\frac{40e^{8\{(x_1-0.5)^2+(x_2-0.5)^2\}}}{e^{8\{(x_1-0.2)^2+(x_2-0.7)^2\}} + e^{8\{(x_1-0.7)^2+(x_2-0.2)^2\}}}$	$x_1, x_2 \in [-1,1]$

Table 1: Functions used for generating synthetic datasets.

Datasets (Train size, Test size)	Uniformly distributed noises			Gaussian noises		
	SVR (C, $\sigma$ )	TSVR (C <sub>1</sub> =C <sub>2</sub> , $\sigma$ )	Our Method (C <sub>1</sub> =C <sub>2</sub> , $\sigma$ )	SVR (C, $\sigma$ )	TSVR (C <sub>1</sub> =C <sub>2</sub> , $\sigma$ )	Our Method (C <sub>1</sub> =C <sub>2</sub> , $\sigma$ )
Function1 (200x1,1000x1)	0.07165 (10 <sup>3</sup> ,2 <sup>0</sup> )	0.04892 (10 <sup>0</sup> ,2 <sup>0</sup> )	<b>0.04626</b> (10 <sup>1</sup> ,2 <sup>0</sup> )	0.1021 (10 <sup>3</sup> ,2 <sup>0</sup> )	0.07694 (10 <sup>2</sup> ,2 <sup>0</sup> )	<b>0.07085</b> (10 <sup>2</sup> ,2 <sup>0</sup> )
Function2 (200x1,1000x1)	0.06283 (10 <sup>3</sup> ,2 <sup>1</sup> )	<b>0.04631</b> (10 <sup>2</sup> ,2 <sup>0</sup> )	0.04634 (10 <sup>2</sup> ,2 <sup>0</sup> )	0.13271 (10 <sup>0</sup> ,2 <sup>0</sup> )	0.08354 (10 <sup>-1</sup> ,2 <sup>0</sup> )	<b>0.08337</b> (10 <sup>-3</sup> ,2 <sup>0</sup> )
function3 (200x5,1000x5)	<b>0.04689</b> (10 <sup>3</sup> ,2 <sup>1</sup> )	0.05968 (10 <sup>3</sup> ,2 <sup>1</sup> )	0.05736 (10 <sup>2</sup> ,2 <sup>1</sup> )	<b>0.05289</b> (10 <sup>3</sup> ,2 <sup>2</sup> )	0.07818 (10 <sup>1</sup> ,2 <sup>1</sup> )	0.06996 (10 <sup>1</sup> ,2 <sup>1</sup> )
function4 (200x2,1000x2)	0.0216 (10 <sup>3</sup> ,2 <sup>0</sup> )	0.01206 (10 <sup>1</sup> ,2 <sup>0</sup> )	<b>0.01107</b> (10 <sup>2</sup> ,2 <sup>0</sup> )	0.05202 (10 <sup>3</sup> ,2 <sup>1</sup> )	0.03384 (10 <sup>-1</sup> ,2 <sup>0</sup> )	<b>0.03271</b> (10 <sup>0</sup> ,2 <sup>0</sup> )
function5 (200x2,1000x2)	0.06159 (10 <sup>1</sup> ,2 <sup>-3</sup> )	<b>0.02062</b> (10 <sup>2</sup> ,2 <sup>-3</sup> )	0.02068 (10 <sup>3</sup> ,2 <sup>-3</sup> )	0.07926 (10 <sup>0</sup> ,2 <sup>-4</sup> )	<b>0.04721</b> (10 <sup>0</sup> ,2 <sup>-2</sup> )	0.04916 (10 <sup>0</sup> ,2 <sup>-2</sup> )
function6 (200x2,1000x2)	0.09457 (10 <sup>3</sup> ,2 <sup>-2</sup> )	0.0228 (10 <sup>1</sup> ,2 <sup>-1</sup> )	<b>0.02243</b> (10 <sup>3</sup> ,2 <sup>-1</sup> )	<b>0.03978</b> (10 <sup>3</sup> ,2 <sup>-1</sup> )	0.04168 (10 <sup>1</sup> ,2 <sup>0</sup> )	0.04225 (10 <sup>1</sup> ,2 <sup>0</sup> )

Table 2: Performance comparison of our method with SVR and TSVR on synthetic datasets for uniform and Gaussian additive noises. RMSE was used for comparison. Gaussian kernel was employed. Bold type shows the best result.

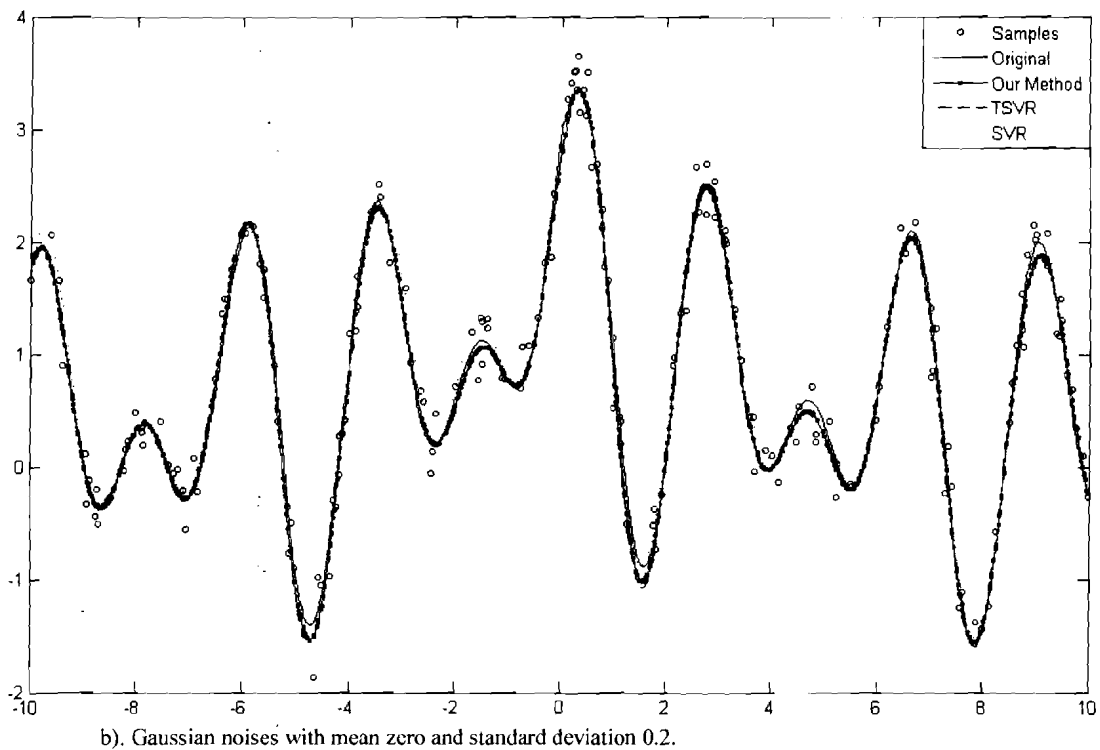
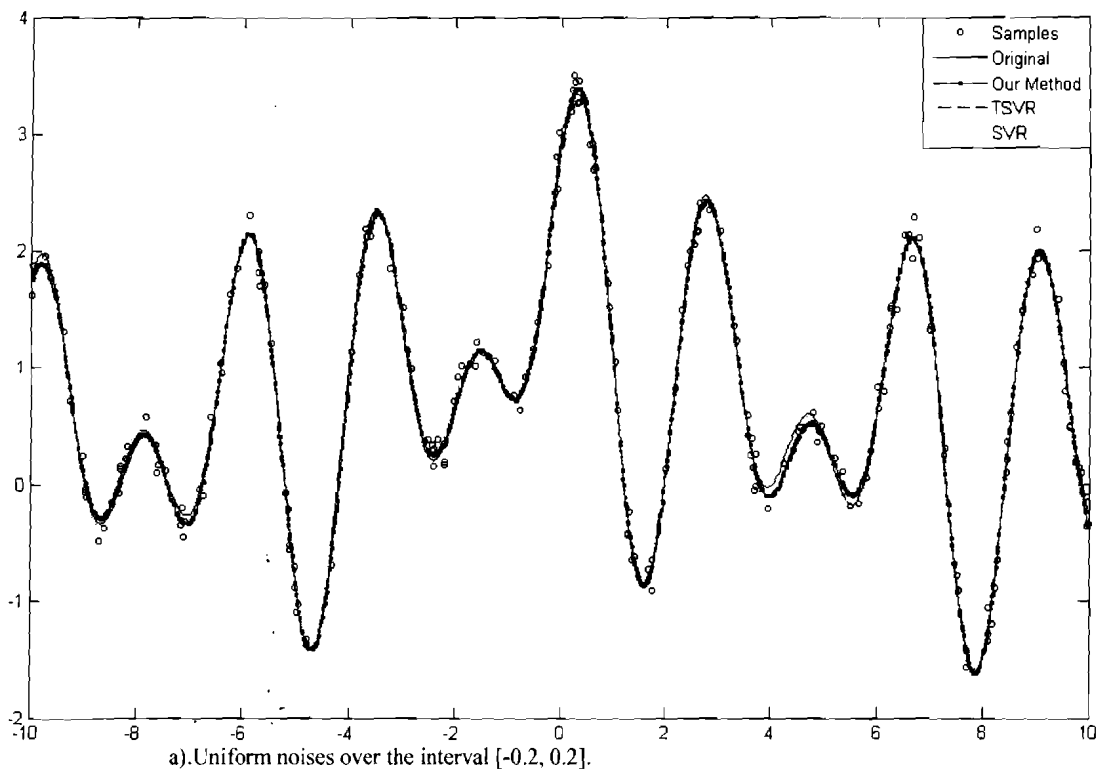


Figure 6: Results of approximation of  $\frac{4}{|x|+2} + \cos(2x) + \sin(3x)$  with our proposed method, SVR and TSVR on noisy input samples. Gaussian kernel was employed.

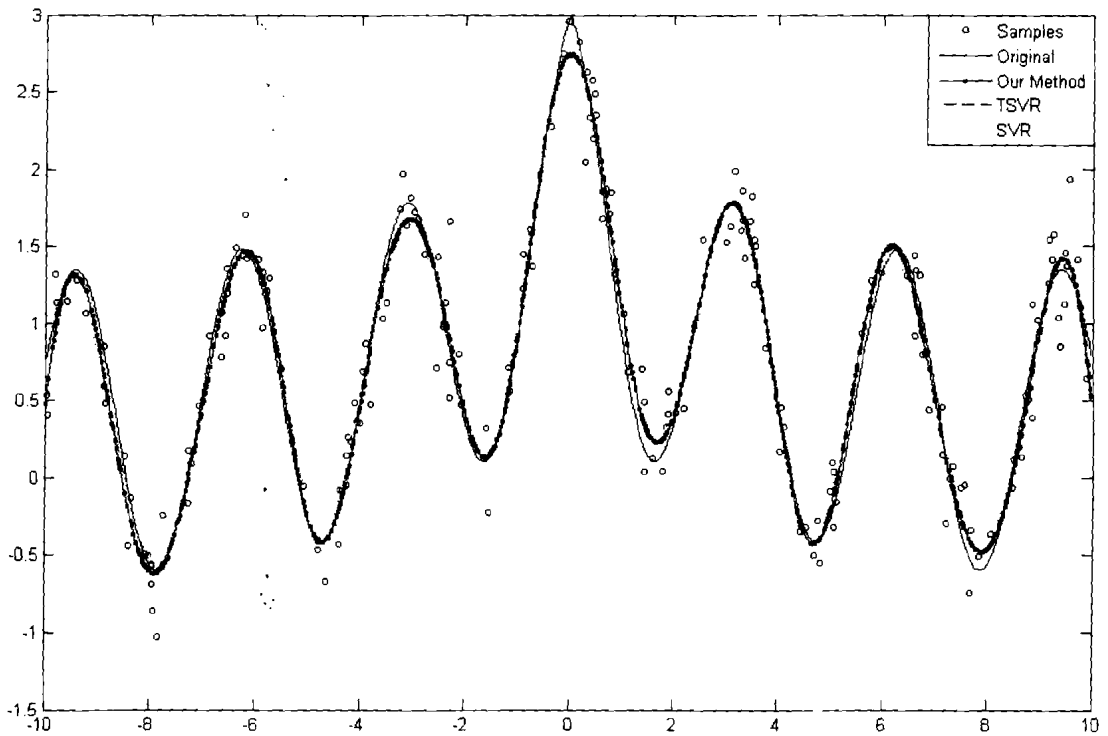
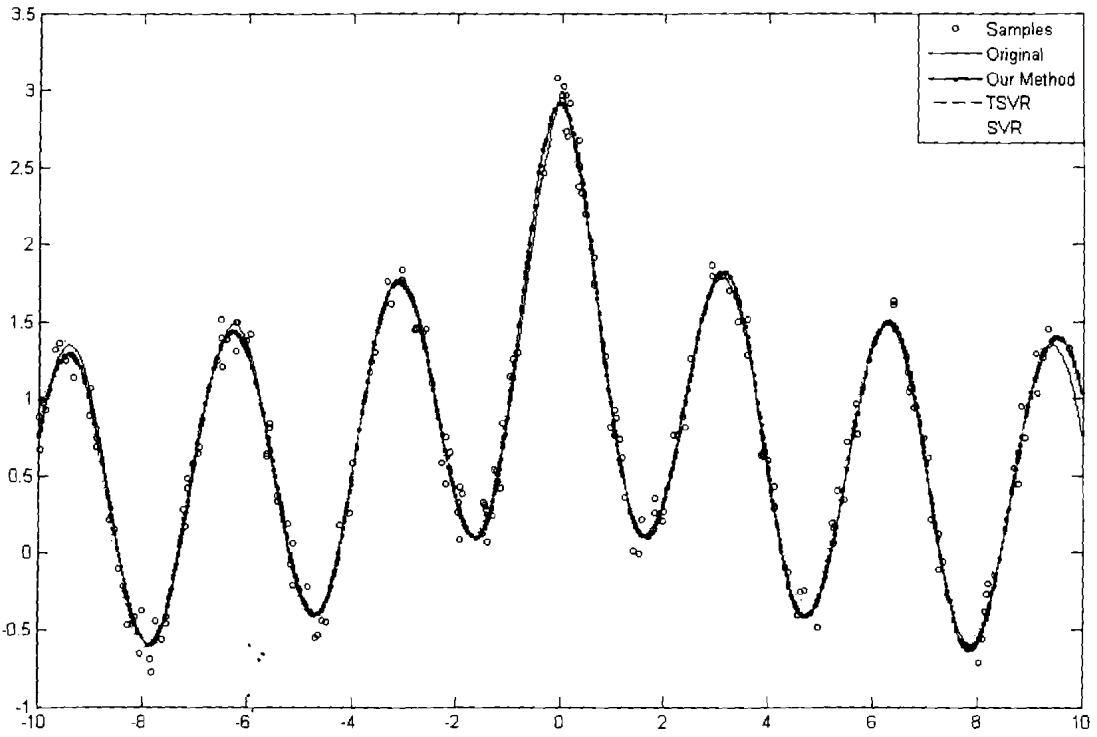


Figure 7: Results of approximation of  $\frac{4}{|x|+2} + \cos(2x)$  with our proposed method, SVR and TSVR on noisy input samples. Gaussian kernel was employed.



### 4.3. Experiments on real-world benchmark datasets

In order to further experiment the effectiveness of method, we compare the proposed method with TSVR, and SVR on eighteen real-world data sets. As the first real-world example, we considered the Box and Jenkins gas furnace data (Box and Jenkins, 1976). It consists of 296 input-output pairs of points of the form  $(u(t), y(t))$  where  $u(t)$  is input gas flow rate whose output  $y(t)$  is the  $CO_2$  concentration from the gas furnace. The output  $y(t)$  is predicted based on 6 attributes taken to be of the form  $x(t) = (y(t-1), y(t-2), y(t-3), u(t-1), u(t-2), u(t-3))$ , (Wang et al., 2006). This makes the total number of samples become 293 where each sample is of the form  $(x(t), y(t))$ . The even samples were selected for training whereas the odd samples were taken for testing. The performance of the proposed method on the training and test sets were shown in Fig.8a and Fig.8b respectively.

As examples of financial time series, we considered the datasets of Google, IBM, Intel, RedHat, MS stock and S&P 500. They were taken from the website: <http://dailyfinance.com>. In all the examples, 755 closing prices starting from 01-01-2006 to 31-12-2008 were taken. Since we used five previous values to predict the current value, 750 samples in total were obtained. Among them the first 150 samples were used for training and the remaining for testing.

In addition, we compared implicit LTSVR with the standard SVR and TSVR on few more well-known time series datasets. In all of them considered, five previous values were used to predict the current value.

The Sunspot dataset is taken from <http://www.bme.ogi.edu/~ericwan/data.html>. It consists of 295 yearly readings from the year 1700 to 1994. As five previous values are used to predict the current value, 290 samples were obtained. We used 100 samples for training and the remaining samples for testing. The SantaFe-A time series dataset is taken from <http://www-psych.stanford.edu/~andreas/Time-Series/SantaFe.html>. It consists of 1000 values from which we get 995 samples in total. Among them 300 were chosen for training and the remaining for testing.

We performed our experiments on two time series generated by the Mackey-Glass time delay differential equation (Mukherjee et al, 1997) given by

$$\frac{dx(t)}{dt} = -bx(t) + a \frac{x(t-\tau)}{1+x(t-\tau)^{10}},$$

where the parameters  $a$ ,  $b$  and time delay  $\tau$  were taken as:  $a = 0.2, b = 0.1$  and  $\tau = 17, 30$ . They were obtained from <http://www.bme.ogi.edu/~ericwan/data.html>. The first 400 number of samples were considered for training and the remaining 1095 for testing. In this work, we denote the time series corresponding to  $\tau = 17$  and  $\tau = 30$  by  $MG_{17}$  and  $MG_{30}$  respectively.

Consider the Lorenz differential equation (Casdagli 1989) given by

$$\dot{x} = \rho(y - x), \dot{y} = rx - y - xz \text{ and } \dot{z} = xy - bz,$$

where  $\rho, r$  and  $b$  are input parameters. By taking differential sampling rates as:  $\tau = 0.05$  and  $\tau = 0.2$  and the parameter values as  $\rho = 10, r = 28, b = 8/3$  the time series datasets  $Lorenz_{0.05}$  and  $Lorenz_{0.2}$  were generated. They consist of 30000 number of values. To avoid the initial transients the first 1000 of them were discarded. The next 3000 of them were taken for our experiment. With five previous values being used to predict the current value, we get 2995 number of samples. Among them the first 400 samples were considered for training and the remaining samples for testing.

Finally, five commonly used bench mark datasets- Machine CPU, Bodyfat, concrete CS, Abalone and Kin-fh- were considered in this study for testing the performance of the implicit LTSVR.

The Machine CPU dataset consisting of 209 samples having 7 continuous features is taken from UCI repository (Murphy and Aha, 1992). The first 100 samples were chosen for training and the remaining for testing. Bodyfat is a well-known dataset available in the Statlib collection: <http://lib.stat.cmu.edu/datasets>. It consists of 252 samples having from the body density values. The first 150 samples were taken for training and the remaining for testing.

In all the previos real-world examples, the original data is normalized with mean and satndard deviation equal to 0 and 1 respectively. However, in the following three datasets, the original data is normalized in the following manner:

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

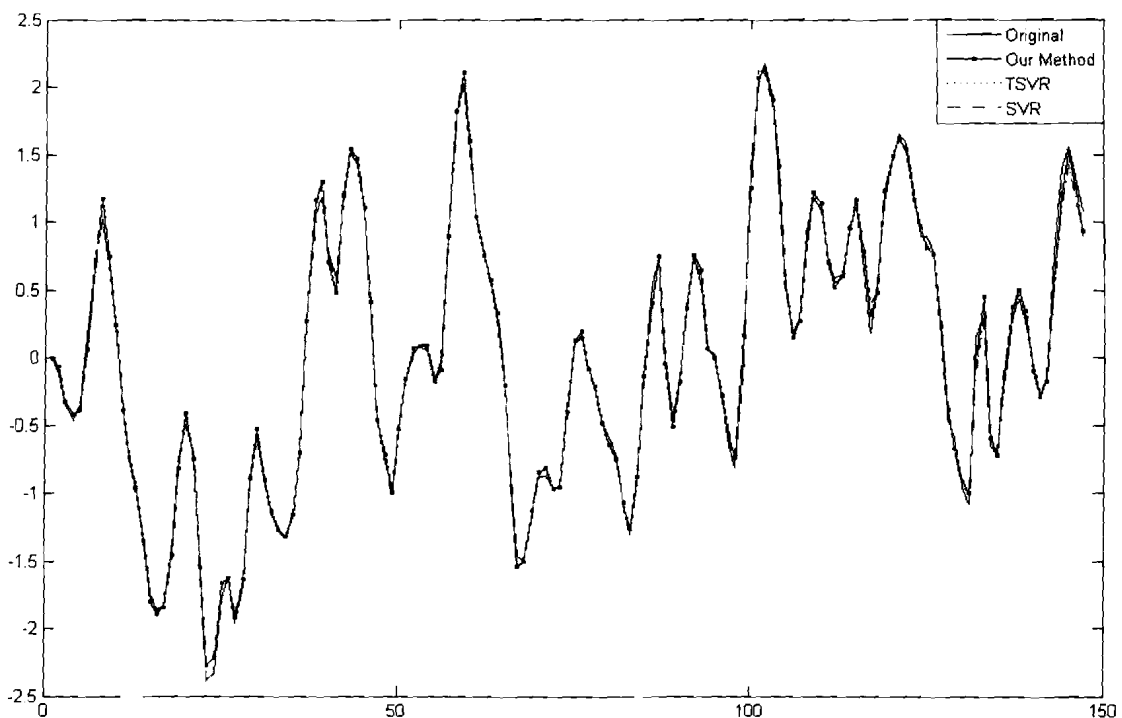
where  $x_{ij}$  is the  $(i,j)$ -th element of the input matrix  $A$ ,  $\bar{x}_{ij}$  is its corresponding normalized value and  $x_j^{\min} = \min_{i=1}^m(x_{ij})$  and  $x_j^{\max} = \max_{i=1}^m(x_{ij})$  denote the minimum and maximum values, respectively, of the  $j$ -th feature of  $A$ .

The concrete CS dataset (Murphy and Aha, 1992) contains 1030 samples having 8 features. The first 700 samples were used for training and the remaining for testing. The goal of Abalone dataset (Murphy and Aha, 1992) is the prediction of the ages of the abalones using 8 physical measurements as their features. In our experiment we have chosen 1000 samples for training and the remaining 3177 samples for testing. The Kin-fh dataset (DELVE, 2005) represents a realistic simulation of the forward dynamics of eight link all revolute robot arm. Its goal is the prediction of the end-effector from a target with 32 features. The first 1000 samples were chosen for training and the remaining 7192 samples for testing.

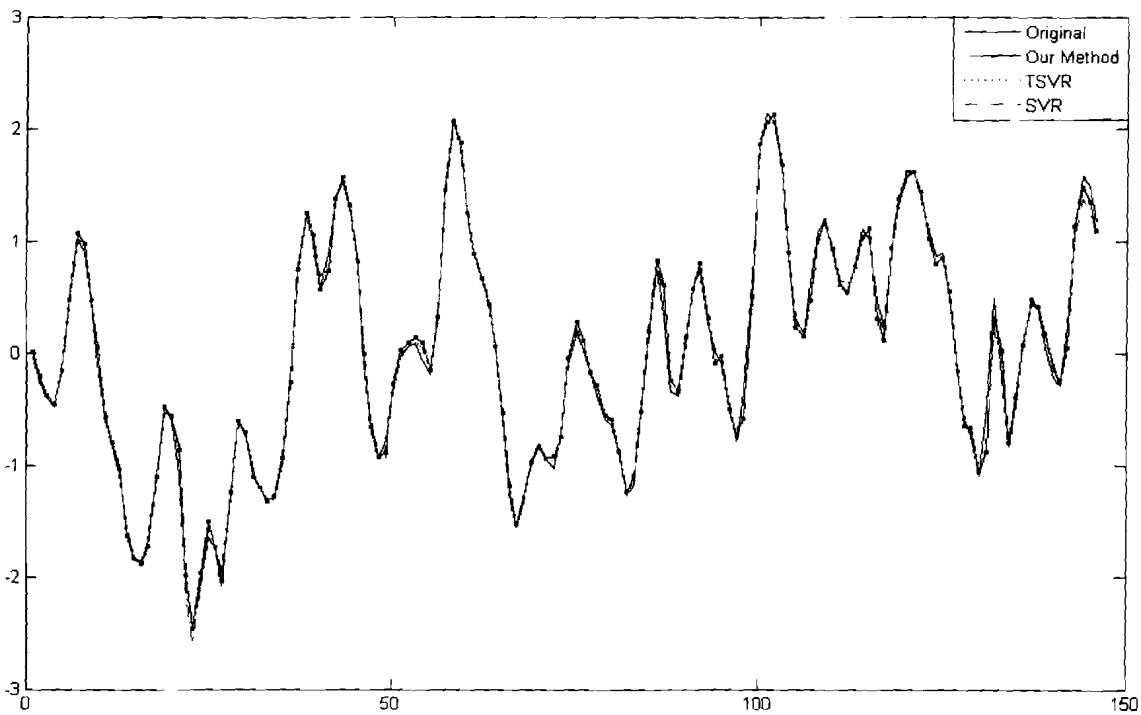
The tenfold numerical results of the real-world datasets by the standard SVR, TSVR and the implicit LTSVR along with number of training and test samples chosen, the number of attributes and the parameter values were summarized in Table 3. From Table 2 and Table 3, it can be observed that implicit LTSVR is an effective and powerful tool for solving regression problems.

Datasets (Train size, Test size)	SVR ( $C, \sigma$ )	TSVR ( $C_1=C_2, \sigma$ )	Our Method ( $C_1=C_2, \sigma$ )
Gas furnace (147X6,146X6)	0.08614 ( $10^3, 2^{-9}$ )	0.08423 ( $10^3, 2^2$ )	<b>0.08422</b> ( $10^3, 2^2$ )
Google (150X5,600X5)	0.1381 ( $10^1, 2^4$ )	<b>0.1359</b> ( $10^{-1}, 2^4$ )	0.1378 ( $10^{-1}, 2^4$ )
IBM (150X5,600X5)	0.1494 ( $10^1, 2^3$ )	0.1364 ( $10^{-2}, 2^3$ )	<b>0.1362</b> ( $10^{-3}, 2^3$ )
Intel (150X5,600X5)	0.1541 ( $10^3, 2^6$ )	0.1566 ( $10^0, 2^4$ )	<b>0.1506</b> ( $10^1, 2^4$ )
Red Hat (150X5,600X5)	0.2031 ( $10^1, 2^4$ )	0.17911 ( $10^0, 2^4$ )	<b>0.17907</b> ( $10^0, 2^4$ )
MS stock (150X5,600X5)	0.2180 ( $10^1, 2^4$ )	0.2183 ( $10^0, 2^3$ )	<b>0.2166</b> ( $10^{-1}, 2^3$ )
S&P500 (150X5,600X5)	<b>0.1757</b> ( $10^3, 2^5$ )	0.2606 ( $10^{-3}, 2^3$ )	0.2607 ( $10^{-3}, 2^3$ )
Sunspots (100X5,190X5)	0.3778 ( $10^1, 2^2$ )	<b>0.3742</b> ( $10^0, 2^3$ )	0.3782 ( $10^0, 2^3$ )
SantafeA (300X5,695X5)	<b>0.1751</b> ( $10^1, 2^1$ )	0.1802 ( $10^0, 2^1$ )	0.1905 ( $10^3, 2^0$ )
MG <sub>17</sub> (400X5,1095X5)	<b>0.0123</b> ( $10^3, 2^2$ )	0.0151 ( $10^{-1}, 2^0$ )	0.0149 ( $10^3, 2^0$ )
MG <sub>30</sub> (400X5,1095X5)	<b>0.1006</b> ( $10^1, 2^1$ )	0.1025 ( $10^{-2}, 2^0$ )	0.1030 ( $10^{-3}, 2^0$ )
Lorenz <sub>0.05</sub> (400X5,2595X5)	<b>0.0152</b> ( $10^3, 2^5$ )	0.02202 ( $10^2, 2^1$ )	0.02189 ( $10^3, 2^1$ )
Lorenz <sub>0.2</sub> (400X5,2595X5)	<b>0.0234</b> ( $10^3, 2^6$ )	0.0858 ( $10^{-1}, 2^1$ )	0.0808 ( $10^1, 2^1$ )
Machine CPU (100X7,109X7)	<b>0.1215</b> ( $10^3, 2^5$ )	0.1537 ( $10^0, 2^3$ )	0.1533 ( $10^3, 2^3$ )
Bodyfat (150X14,102X14)	<b>0.0594</b> ( $10^3, 2^8$ )	0.0787 ( $10^0, 2^5$ )	0.1057 ( $10^{-1}, 2^5$ )
Concrete CS (700X8,330X8)	0.1770 ( $10^1, 2^{-1}$ )	<b>0.16061</b> ( $10^{-2}, 2^1$ )	0.16062 ( $10^{-3}, 2^1$ )
Abalone (1000X8,3177X8)	0.2097 ( $10^1, 2^0$ )	0.1189 ( $10^{-1}, 2^1$ )	<b>0.1174</b> ( $10^0, 2^1$ )
Kin-fh (1000X32,7192X32)	0.0960 ( $10^{-1}, 2^1$ )	0.09523 ( $10^{-3}, 2^3$ )	<b>0.09522</b> ( $10^{-3}, 2^3$ )

Table 3: Performance comparison of our proposed method with SVR and TSVR on real world datasets. RMSE was used for comparison. Gaussian kernel was employed. Bold type shows the best result.



a). Prediction over the training set



b). Prediction over the test set

Figure 8: Results of comparison for gas furnace data of Box-Jenkins. Gaussian kernel was used.

## Conclusion and Future Work

A new implicit Lagrangian twin SVR in its dual was proposed as a pair of unconstrained minimization problems and their solutions were obtained by Newton method. The algorithm is very simple to implement and no specialized optimization software is required. Numerical experiments were performed on a number of interesting synthetic and real-world datasets. Comparison of results with SVR and twin SVR clearly demonstrates the effectiveness and suitability of the proposed method. As an extension of the proposed work, the study of the smoothing technique for the implicit twin SVR will be explored.

## Appendix

### MATLAB code for implicit LTSVR using Newton's method

```
%clear;
%clc;
function [RMSE,final,iter1,iter2] = NTSVM_nonlinear...
    (Data,Data_test,mew,nu,epsilon,precision)
%-----
% a has complete input data.
% mew is kernel parameter.
% precision = 10^-5.
[m,n]=size(Data);
a=Data(:,1:n-1);    %input data
[input_row,input_col]=size(a);
y1=Data(:,n);

alpha1=100000.;
alpha2=100000.;

% kernel function
mew1=1/(2*mew*mew);
K=zeros(input_row,input_row);
for i=1:input_row
    for j=1:input_row
        nom = norm( [a(i,:) 1] - [a(j,:) 1] );
        K(i,j) = exp( (-1/(2*mew*mew)) * nom * nom );
    end
end

iter1=0;
u1=zeros(input_row,1);
e1=ones(input_row,1);
h1=[K e1];
Q1=speye(input_row)/nu + h1*inv( h1'*h1+0.0001*...
    speye(input_row+1))*h1';
r1=(h1*inv(h1'*h1 + 0.0001*speye(input_row+1))*h1'-...
    speye(input_row))*(y1-epsilon*e1);
hul=-max(((Q1*u1-r1) - alpha1*u1),0)+ Q1*u1-r1;
while norm(hul) > precision && iter1 < 100
    iter1=iter1+1;
    star1=sign(max(((Q1-alpha1*eye(input_row))*u1-
r1),0));
    dhul=sparse((eye(input_row)-diag(star1))*Q1...
        + alpha1*diag(star1));
    delta1=dhul\hul;
    unew1=u1-delta1;
    u1=unew1;
    hul=-max(((Q1*u1-r1)-alpha1*u1),0)+Q1*u1-r1;
end
```

```

iter2=0;
u2=zeros(input_row,1);
e2=ones(input_row,1);
h2=[K e2];
y2=y1;
Q2=speye(input_row)/nu + h2*inv(h2'*h2 + 0.0001*...
    speye(input_row+1))*h2';
r2=(-h2*inv(h2'*h2 + 0.0001*speye(input_row+1))*h2'...
    + speye(input_row))*(y2+epsilon*e2);
hu2=-max(((Q2*u2-r2) - alpha2*u2),0) + Q2*u2-r2;
while norm(hu2) > precision && iter2 < 100
    iter2=iter2+1
    star2=sign(max(((Q2-alpha2*eye(input_row))*u2-
r2),0));
    dhu2=sparse((eye(input_row)-diag(star2))*Q2 ...
        + alpha2*diag(star2));
    delta2=dhu2\hu2;
    unew2=u2-delta2;
    u2=unew2;
    hu2=-max(((Q2*u2-r2)-alpha2*u2),0)+Q2*u2-r2;
end
% Testing part is done here
p_test=Data_test(:,1:n-1);
[no_test,no_col_test]=size(p_test);
y_test=Data_test(:,n); % Actual y value
e = ones(no_test,1);
% Kernel function
K_test=zeros(no_test,input_row);
for i=1:no_test
    K_test(i) = 0.;
    for j=1:input_row
        nom = norm([p_test(i,:) 1] - [a(j,:) 1] );
        K_test(i,j) = exp( (-1/(2*mew*mew)) * nom * nom );
    end
end
end
F1 = [K_test e]*inv(h1'*h1 + 0.0001*speye(input_row+1))...
    *h1'*(y1-e1*epsilon - u1);
F2 = [K_test e]*inv(h2'*h2 + 0.0001*speye(input_row+1))...
    *h2'*(y2+e2*epsilon + u2);
final=(F1+F2)/2; % final regressor
% Calculating the Root Mean Square error(RMSE)
RMSE = norm(y_test-final)/sqrt(no_test);

```



## References

- Balasundaram, S., & Kapil (2010). On Lagrangian support Vector Regression, *Expert Systems with Applications*, 37, 8784-8792.
- Balasundaram, S., & Rampal Singh (2010). On finite Newton method for support vector regression, *Neural Computing & Applications*, 19,967-977.
- Bao Y.-K., Liu Z.-T., Guo L., & Wang W. (2005). "Forecasting Stock Composite Index by Fuzzy Support Vector Machines Regression", *Proc. of 4<sup>th</sup> Int. Conf. on the Machine Learning and Cybernetics*, Guangzhou, pp. 18-21.
- Box, G.E.P., & Jenkins, G.M. (1976). *Time series analysis: Forecasting and Control*, Holden-Day, San Francisco.
- Brockwell, P.J., & Davis R.A. (2002). *Introduction to Time Series Forecasting*, 2<sup>nd</sup> Ed. Springer-Verlag, Berlin.
- Brown M.P.S., Grundy W.N., & Lin, D. (2000). Knowledge-based analysis of microarray gene expression data using support vector machine, *Proceedings of the National Academy of Sciences of USA*, 97(1), 262-267.
- Burges C.J.C. (1998). *A Tutorial on Support Vector Machines for Pattern Recognition*. Data Mining and Knowledge Discovery, 2(2), pp.121-167.
- Cao L.J. (2003). "Support Vector Machines Experts for Time Series Forecasting", *Neurocomputing*, Vol. 51, pp. 321-339.
- Casdagli, M. (1989). Nonlinear prediction of chaotic time series, *Physica D*, 35, 335-356.
- Chang, C.C., & Lin. C.J. (2001). LIBSVM: A Library for Support Vector Machines, <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- Chen, S., & Wang, M. (2005). Seeking multi-threshold directly from support vectors for image segmentation, *Neurocomputing*, 67, 335-344.
- Chen X., Yang J., Liang J., & Ye, Q. (2011). Smooth twin support vector regression, *Neural Computing & Applications*, doi 10.1007/s00521-010-0454-9
- Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines and kernel based learning methods*, Cambridge University Press.
- DELVE, (2005). Data for Evaluating Learning in Valid Experiments, <http://www.cs.toronto.edu/~delve/data>.
- Demiriz A., Bennett K., Breneman C. & Embrechts M. (2001). "Support Vector Machine Regression in Chemometrics", *Computing Science and Statistics*.
- Dibike Y.B., Velickov S., & Solomatine D. (2000). "Support Vector Machines: Review and Applications in Civil Engineering", in *AI Methods in Civil Engineering Applications*, Schleider et al., (Eds.), Cottbus, pp.45-58.
- Fung, G., & Mangasarian, O.L. (2003). Finite Newton method for Lagrangian support vector machine, *Neurocomputing*, 55, 39-55.
- Hiriart-Urruty, J.-B., Strodriot, J.J., & Nguyen, H. (1984). Generalized Hessian matrix and second order optimality conditions for problems with  $C^{L1}$  data, *Applied Mathematics and Optimization*, 11, 43-56.

- Jayadeva, Khemchandani R., & Chandra, S. (2007). Twin support vector machines for pattern classification, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(5), 905-910.
- Kumar M.A., & Gopal, M. (2008). Application of smoothing technique on twin support vector machines, *Pattern Recognition Letters*, 29, 1842-1848.
- Kumar M.A., & Gopal, M. (2009). Least squares twin support vector machines for pattern classification, *Expert system with Applications*, 36, 7535-7543.
- Lawrence & Jeanette (1994). *Introduction to Neural Networks*, California Scientific Software Press. ISBN 1-883157-00-5.
- Lee, Y.J., Hsieh, W.-F., & Huang, C.-M. (2005).  $\epsilon$ -SSVR: A smooth support vector machine for  $\epsilon$ -insensitive regression, *IEEE Transactions on Knowledge and Data Engineering*, 17(5), 678-685.
- Li, G., Wen, C., Huang, G.-B & Chen, Y. (2011). Error tolerance based support vector machine for regression, *Neurocomputing*, 74(5), 771-782.
- Mangasarian, O.L. (1994). *Nonlinear programming*, SIAM Philadelphia, PA.
- Mangasarian, O.L., & Musicant, D.R. (2001). Lagrangian support vector machines, *Journal of Machine Learning Research*, 1, 161-177.
- Mangasarian, O.L., & Solodov, M.V. (1993). Nonlinear complementarity as unconstrained and constrained minimization, *Mathematical Programming, Series B*, 62, 277-297.
- Mangasarian, O.L., & Wild, F.W. (2006). Multisurface proximal support vector classification via generalized eigenvalues, *IEEE Transactions on Pattern Analysis and machine Intelligence* 28(1), pp.69-74.
- Mukherjee, S., Osuna, E., & Girosi, F. (1997). Nonlinear prediction of chaotic time series using support vector machines, in: *NNSP'97: Neural Networks for Signal Processing VII: in Proceedings of IEEE Signal Processing Society Workshop, Amelia Island, FL, USA*, 511-520.
- Muller, K.R., Smola, A.J., Ratsch, G., Schölkopf, B., & Kohlmorgen, J. (1999). Using support vector machines for time series prediction, in: B.Schölkopf, C.J.C.Burges, A.J.Smola (Eds.), *Advances in Kernel methods- Support Vector Learning*, MIT Press, Cambridge, MA, 243-254.
- Murphy, P.M., & Aha, D.W. (1992). UCI repository of machine learning databases. University of California, Irvine. <http://www.ics.uci.edu/~mllearn>
- Musicant, D.R., & Feinberg, A. (2004). Active set support vector regression, *IEEE Transactions on Neural Networks*, 15(2), 268-275.
- Osuna, E., Freund, R., & Girosi, F. (1997). Training support vector machines: An application to face detection, in *proceedings of Computer Vision and Pattern Recognition*, 130-136.
- Peng, X. (2010). TSVR: An efficient Twin Support Vector Machine for regression, *Neural Networks*, 23(3), 365-372.
- Suykens, J.A.K., Vandewalle, J., B.d. (2001). Optimal control by least squares *support vector machines*, *Neural Networks*, 14(1), 23-25.
- Tay F.E.H., & Cao L.J. (2001). "Application of Support Vector Machines in Financial Time Series with Forecasting", *Omega*, Vol.29, No.4, pp.309-317.
- Tong Q., Zheng H. & Wang X. (2005). "Gene Prediction Based on the Statistical Combination and the Classification in terms of Gene Characteristics", *Int. Conf. on Neural Networks and Brain*, Vol.2, pp.673-677.

Vapnik, V.N. (2000). *The nature of statistical learning theory*, 2<sup>nd</sup> Edition, Springer, New York.

Wang, X.X, Chen, S., Lowe, D., & Harris, C.J. (2006). Sparse support vector regression based on orthogonal forward selection for generalized kernel model, *Neurocomputing*, 70, 462-474.

Zhong, P., Xu, Y., & Zhao, Y. (2011). Training twin support vector regression via linear programming, *Neural computing & Applications*, doi 10.1007/s00521-011-0526-6

