Filomat 31:7 (2017), 2123–2142
DOI 10.2298/FIL1707123T

Published by Faculty of Sciences and Mathematics,
University of Niš, Serbia
Available at: http://www.pmf.ni.ac.rs/filomat

Linear Programming Twin Support Vector Regression

M. Tanveer[a]

[a] *Discipline of Mathematics, Indian Institute of Technology Indore, Indore 453 552 INDIA*

Abstract. In this paper, a new linear programming formulation of a 1-norm twin support vector regression is proposed whose solution is obtained by solving a pair of dual exterior penalty problems as unconstrained minimization problems using Newton method. The idea of our formulation is to reformulate TSVR as a strongly convex problem by incorporated regularization technique and then derive a new 1-norm linear programming formulation for TSVR to improve robustness and sparsity. Our approach has the advantage that a pair of matrix equation of order equals to the number of input examples is solved at each iteration of the algorithm. The algorithm converges from any starting point and can be easily implemented in MATLAB without using any optimization packages. The efficiency of the proposed method is demonstrated by experimental results on a number of interesting synthetic and real-world datasets.

1. Introduction

Support Vector Machine (SVM) introduced by Vapnik and coworkers [11, 45, 46] is an excellent kernel-based machine learning tool for binary classification and regression. It has been successfully applied to many important class of problems like face detection [32], text categorization [23], gene selection [20], brain-computer interface [14], image segmentation [8] and time series analysis [29, 30]. Since it exhibits better performance in a wide variety of real-world applications in comparison to other popular machine learning methods like artificial neural networks (ANNs), it becomes a powerful paradigm for classification and regression.

Recently, variants of SVM wherein two non-parallel planes are constructed become popular in the literature [22, 28, 33]. The generalized eigenvalue proximal SVM (GEPSVM) proposed by Mangasarian and Wild [28] seeks two non-parallel planes in which each one of them will be closest to its own class of data but at the same time as far away as possible from the other class of data. They are found to be the eigenvectors corresponding to the smallest eigenvalues of two related generalized eigenvalue problems. In the sprit of GEPSVM, Jayadeva et al. [22] proposed twin SVM (TWSVM) for binary classification wherein two non-parallel planes are constructed by solving a pair of quadratic programming problems (QPPs) of smaller size than a single large one as in SVM. Experimental results show the effectiveness of TWSVM over SVM and GEPSVM [22]. TWSVM takes $O\left(\frac{1}{4}m^3\right)$ operations which is 1/4 of the standard SVM, whereas

2010 *Mathematics Subject Classification*. Primary 60K05; Secondary 49M15
Keywords. Linear programming; 1-norm support vector machines; Newton method; Twin support vector regression.
Received: 18 February 2014; Accepted: 27 July 2014
Communicated by Predrag Stanimirović
Acknowledgment: The author would like to express his sincere gratitude to Professor S. Balasundaram for his help during the preparation of this manuscript. This research was supported by Council of Scientific & Industrial Research (CSIR), Government of India.
Email address: tanveergouri@gmail.com; mtanveer@iiti.ac.in (M. Tanveer)

GEPSVM takes $O\left(\frac{1}{4}n^3\right)$. Here, m is the number of training examples; n is the dimensionality and $m >> n$ [22, 24]. It is obvious that GEPSVM is far faster than TWSVM. Some scholars proposed variants of TWSVM to reduce the time complexity and keep the effectiveness of TWSVM, see [1–4, 9, 10, 18, 24, 34, 37–43, 47–49].

Inspired by the study of TWSVM, Peng [33] proposed a non-parallel plane regressor called twin support vector regression (TSVR) in which a pair of nonparallel functions corresponding to the $\epsilon-$insensitive down- and up- bounds of the unknown regressor are constructed. In fact, they are obtained by solving a pair of dual QPPs of smaller size than a single large one as in the standard support vector regression (SVR). Experimental results show the generalization performance of TSVR over SVR. It is well known that one significant advantage of SVR is the implementation of the structural risk minimization principle. However, only empirical risk is considered in the primal problems of TSVR due to its complex structure and thus may incur overfitting and suboptimal in some cases. The formulation of TSVR is sensitive to outliers because of the well known outlier-sensitiveness disadvantage of the squared loss function [44]. Furthermore, the solution of TSVR is not capable of generating very sparse solution.

To overcome the shortcomings described above and motivation from the work of Mangasarian [26], recently Tanveer [40–42] proposed novel robust and sparse linear programming twin support vector machines for classification problems. The significant advantage of this formulation is the implementation of structural risk minimization principle and the solution of two modified unconstrained minimization problems reduces to solving just two systems of linear equations as opposed to solving two quadratic programming problems in TWSVM, TBSVM and TSVR, which leads to extremely simple and fast algorithm. In the spirit of Tanveer [40–42], we propose in this paper a new linear programming formulation of 1-norm TSVR to improve robustness and sparsity, whose solution is obtained, by solving a pair of exterior penalty problems in dual as unconstrained optimization problems using Newton-Armijo algorithm. In order to avoid the use of LP packages, illuminated by the exterior penalty (EP) theory [15, 17, 26], we first reformulate the linear programming (LP) problem in the dual space and then construct a completely unconstrained quadratic convex optimization problem. The dual LP formulation can be easily solved using a generalized Newton method. Since the objective functions are not twice differentiable, both the generalized Hessian [16] and smoothing approaches [25] are considered. To demonstrate the effectiveness of the proposed method, we performed numerical experiments on a number of interesting synthetic and real-world datasets and compared their results with other SVRs.

In this work, all vectors are taken as column vectors. The inner product of two vectors x, y in the $n-$dimensional real space R^n, will be denoted by: $x^t y$, where x^t is the transpose of x. Whenever x is orthogonal to y, we write $x \perp y$. For $x = (x_1, x_2, ..., x_n)^t \in R^n$, the plus function x_+ is defined as: $(x_+)_i = \max\{0, x_i\}$, where $i = 1, 2, ..., n$. Further we define the step function x_* as: $(x_*)_i = 1$ for $x_i > 0, (x_*)_i = 0$ for $x_i < 0$ and $(x_*)_i = 0.5$ when $x_i = 0$. The 2-norm of a vector x and a matrix Q will be denoted by $\|x\|$ and $\|Q\|$ respectively. For simplicity, we drop the 2 from $\|x\|_2$. Also we denote the diagonal matrix of order n whose diagonal elements become the components of the vector $x \in R^n$ by $diag(x)$. The identity matrix of appropriate size is denoted by I and the column vector of ones of dimension m by e. If f is a real valued function of the variable $x = (x_1, x_2, ..., x_n)^t \in R^n$ then the gradient of f is denoted by $\nabla f = (\partial f/\partial x_1, ..., \partial f/\partial x_n)^t$ and the Hessian matrix of f is denoted by $\nabla^2 f = (\partial^2 f/\partial x_i \partial x_j)_{i,j=1,2,...,n}$.

The paper is organized as follows. Section 2 dwells briefly SVR and TSVR. In Section 3, the proposed LPTSVR problem is formulated and its solution is obtained by considering a pair of exterior penalty problems in dual as unconstrained optimization problems solved by Newton- Armijo algorithm. Numerical experiments are performed and their results are compared with other SVRs in Section 4. Finally we conclude our work in Section 5.

2. Background

In this section, we briefly outline the standard SVR and twin SVR formulations. For a more detailed description, the interested reader is referred to [12, 33, 46].

Let a training set of examples $\{(x_i, y_i)\}_{i=1,2,...,m}$ be given where for each input $x_i \in R^n$, its corresponding observed value be $y_i \in R$. Suppose we represent the m input examples in the $n-$dimensional space R^n by a matrix $A \in R^{m\times n}$ whose i^{th} row is taken to be x_i^t and the vector of observed values by $y = (y_1, ..., y_m)^t$.

2.1. Support vector regression formulation

The goal of the standard ε–insensitive error loss SVR problem is in determining the regression estimation function $f : R^n \to R$ of the form:

$$f(x) = w^t\varphi(x) + b, \tag{1}$$

where the nonlinear mapping $\varphi(.)$ takes the input examples into a higher dimensional feature space in which the unknown coefficients are estimated by solving the minimization problem given by [12, 46]:

$$\begin{aligned} \min_{w,b,\xi_1,\xi_2} \ & \frac{1}{2}w^t w + C(e^t\xi_1 + e^t\xi_2) \\ s.t. \quad & y - (\varphi(A)w + eb) \le \varepsilon e + \xi_1, \\ & \varphi(A)w + eb) - y \le \varepsilon e + \xi_2, \\ & \xi_1, \xi_2 \ge 0. \end{aligned} \tag{2}$$

Here ξ_1, $\xi_2 \in R^m$ are vectors of slack variables; $C > 0$, $\varepsilon > 0$ are input parameters and the matrix $\varphi(A) = [\varphi(x_1)^t; \varphi(x_2)^t; ...; \varphi(x_m)^t]$.

By introducing Lagrange multipliers $u_1 = (u_{11}, ..., u_{1m})^t$, $u_2 = (u_{21}, ..., u_{2m})^t \in R^m$ and using the KKT optimality conditions, the nonlinear estimation function $f(.)$, defined as above, can be explicitly obtained to be of the form [12, 46]:

$$f(x) = \sum_{i=1}^{m}(u_{1i} - u_{2i})k(x, x_i) + b, \tag{3}$$

where $k(.,.)$ is a kernel function in which $k(x_i, x_j) = \varphi(x_i)^t\varphi(x_j)$ holds.

2.2. Twin support vector regression (TSVR)

TSVR, introduced by Peng [33], aims at determining two non-parallel functions corresponding to the ε_1-insensitive down- and ε_2-insensitive up- regressors by solving a pair of QPPs of smaller size than solving a single large one as in the classical SVR with the advantage that it is fast and shows good generalization performance.

The linear TSVR algorithm finds the down- and up-bound estimation functions, defined by: for any $x \in R^n$,

$$f_1(x) = w_1^t x + b_1 \quad \text{and} \quad f_2(x) = w_2^t x + b_2, \tag{4}$$

respectively, such that the unknowns $w_1, w_2 \in R^n$ and $b_1, b_2 \in R$ become the solutions of the following pair of QPPs [33]:

$$\begin{aligned} \min_{(w_1,b_1,\xi_1)\in R^{n+1+m}} \ & \frac{1}{2}\|y - \varepsilon_1 e - (Aw_1 + b_1 e)\|^2 + C_1 e^t\xi_1 \\ s.t. \quad & y - (Aw_1 + b_1 e) \ge \varepsilon_1 e - \xi_1, \quad \xi_1 \ge 0 \end{aligned} \tag{5}$$

and

$$\begin{aligned} \min_{(w_2,b_2,\xi_2)\in R^{n+1+m}} \ & \frac{1}{2}\|y + \varepsilon_2 e - (Aw_2 + b_2 e)\|^2 + C_2 e^t\xi_2 \\ s.t. \quad & (Aw_2 + b_2 e) - y \ge \varepsilon_2 e - \xi_2, \ \xi_2 \ge 0, \end{aligned} \tag{6}$$

where $C_1, C_2 > 0$; $\varepsilon_1, \varepsilon_2 > 0$ are input parameters and ξ_1, ξ_2 are vectors of slack variables in R^m.

Once the down- and up-bound regressors are determined, the end regressor $f : R^n \to R$ is obtained by taking their mean, i.e. we have

$$f(x) = \frac{1}{2}(f_1(x) + f_2(x)) \quad \text{for all} \quad x \in R^n. \tag{7}$$

The above approach can be easily extended to nonlinear regressors by following the kernel technique of [27]. For the input matrix $A \in R^{m\times n}$ and a nonlinear kernel function $k(.,.)$ given, let the kernel matrix $K(A, A^t)$ of order m whose (i,j)-th element be defined by: $K(A, A^t)_{ij} = k(x_i, x_j) \in R$ and let $K(x^t, A^t) = (k(x, x_1), ..., k(x, x_m))$ be a row vector in R^m.

The nonlinear TSVR seeks the down- and up- bounds of the form:

$$f_1(x) = K(x^t, A^t)w_1 + b_1 \quad \text{and} \quad f_2(x) = K(x^t, A^t)w_2 + b_2, \tag{8}$$

respectively, which are determined by solving the following pair of QPPs [33]:

$$\begin{aligned} \min_{(w_1,b_1,\xi_1)\in R^{m+1+m}} \ & \frac{1}{2}\|y - \varepsilon_1 e - (K(A, A^t)w_1 + b_1 e)\|^2 + C_1 e^t \xi_1 \\ s.t. \quad & y - (K(A, A^t)w_1 + b_1 e) \geq \varepsilon_1 e - \xi_1, \xi_1 \geq 0 \end{aligned} \tag{9}$$

and

$$\begin{aligned} \min_{(w_2,b_2,\xi_2)\in R^{m+1+m}} \ & \frac{1}{2}\|y + \varepsilon_2 e - (K(A, A^t)w_2 + b_2 e)\|^2 + C_2 e^t \xi_2 \\ s.t. \quad & (K(A, A^t)w_2 + b_2 e) - y \geq \varepsilon_2 e - \xi_2, \xi_2 \geq 0 \end{aligned} \tag{10}$$

and finally the end regressor is obtained using (7). For a detailed study on TSVR, the interested reader is referred to [33].

3. Proposed Linear Programming Twin Support Vector Regression (LPTSVR)

In this section, we first reformulate TSVR as a strongly convex problem by incorporated regularization term and then derive a 1-norm linear programming formulation for TSVR. Before starting our proposed formulation, let us first revisit the model of TSVR. It can be observed that the formulation of TSVR is sensitive to outliers because of the well known outlier-sensitiveness disadvantage of the squared loss function [44]. Furthermore, the objective functions of (5) and (6) are comprised empirical risk and small empirical risk can not ensure good generalization performance since it may suffer from overfitting [46]. Furthermore, the solution of TSVR is not capable of generating very sparse solution.

To overcome the shortcoming described above in TSVR and motivation from the works of Tanveer [40–42], we make some improvements and propose a novel 1-norm linear programming formulation for TSVR to improve robustness and sparsity. In order to avoid the use of LP packages, illuminated by the exterior penalty (EP) theory [15, 17, 26], we first reformulate the LP problem in the dual space and then construct a completely unconstrained quadratic convex optimization problem. The pair of dual LP formulation can be easily solved using a generalized Newton method.

The 2-norm regularized TSVR (RTSVR) formulation can be written as follows:

$$\begin{aligned} \min_{(w_1,b_1,\xi_1)\in R^{n+1+m}} \ & \frac{1}{2}\|y - \varepsilon_1 e - (Aw_1 + b_1 e)\|_2^2 + \frac{C_1}{2}\xi_1^t \xi_1 + \frac{\eta_1}{2}\left\| \begin{bmatrix} w_1 \\ b_1 \end{bmatrix} \right\|_2^2 \\ s.t. \quad & y - (Aw_1 + b_1 e) \geq \varepsilon_1 e - \xi_1, \end{aligned} \tag{11}$$

and

$$\begin{aligned} \min_{(w_2,b_2,\xi_2)\in R^{n+1+m}} \ & \frac{1}{2}\left\|y + \varepsilon_2 e - (Aw_2 + b_2 e)\right\|_2^2 + \frac{C_2}{2}\xi_2^t \xi_2 + \frac{\eta_2}{2}\left\| \begin{bmatrix} w_2 \\ b_2 \end{bmatrix} \right\|_2^2 \\ s.t. \quad & (Aw_2 + b_2 e) - y \geq \varepsilon_2 e - \xi_2. \end{aligned} \tag{12}$$

Notice that the objective functions of RTSVR are regularized via the 2-norm of corresponding coefficients with weights η_1 and η_2 respectively which makes the objective functions strongly convex. Thus, it has a unique global optimal solution. Furthermore, by introducing the regularization terms $\frac{1}{2}\|\begin{bmatrix} w_1 \\ b_1 \end{bmatrix}\|_2^2$ and $\frac{1}{2}\|\begin{bmatrix} w_2 \\ b_2 \end{bmatrix}\|_2^2$ to the framework of TSVR, the regularized TSVR becomes a well-posed model as it can not only help to alleviate overfitting and improve the generalization performance as in conventional SVM but also introduce invertibilty in the dual formulation.

Even though 2-norm distance of residuals is more sensitive to large errors and therefore less robust in comparison to 1-norm and though 1-norm formulation has the advantage of generating sparse solution [6], many methods exist in the literature to solve 2-norm SVR formulated as a QPP but very little on 1-norm linear programming SVR. For the work on the formulation of TSVR as a pair of linear programming problems the interested reader is referred to [18].

Motivated by the study of 1-norm SVM problem formulated as a linear programming optimization problem [26], Tanveer [40–42] recently proposed novel robust and sparse linear programming twin support vector machines for classification problems. In this paper, we are extending the idea for regression problems and proposed a novel linear programming TSVR (LPTSVR) whose solution is obtained, by solving a pair of exterior penalty problems in dual as unconstrained optimization problems using Newton-Armijo algorithm.

We first replaces all the 2-norm terms in RTSVR with 1-norm terms as follows:

$$\begin{aligned} \min_{(w_1,b_1,\xi_1)\in R^{n+1+m}} \; & \|y-\varepsilon_1 e-(Aw_1+b_1e)\|_1 + C_1e^t\xi_1 + \eta_1\|\begin{bmatrix} w_1 \\ b_1 \end{bmatrix}\|_1 \\ s.t. \quad & y-(Aw_1+b_1e) \geq \varepsilon_1 e-\xi_1, \xi_1 \geq 0 \end{aligned} \tag{13}$$

and

$$\begin{aligned} \min_{(w_2,b_2,\xi_2)\in R^{n+1+m}} \; & \|y+\varepsilon_2 e-(Aw_2+b_2e)\|_1 + C_2e^t\xi_2 + \eta_2\|\begin{bmatrix} w_2 \\ b_2 \end{bmatrix}\|_1 \\ s.t. \quad & (Aw_2+b_2e)-y \geq \varepsilon_2 e-\xi_2, \xi_2 \geq 0. \end{aligned} \tag{14}$$

whose solutions $w_1, w_2 \in R^n$ and $b_1, b_2 \in R$ will be used to determine the end regressor (7).
We notice that regularization technique is adopted to convert original TSVR to a well posed problem. Furthermore, 1-norm error loss function and 1-norm regularization are used to introduce robustness and sparseness simultaneously.

Following the approach of Mangasarian [26], we will obtain the solutions of (13) and (14) in two steps: (i). formulate the pair (13) and (14) as a pair of linear programming problems (LPPs); (ii). obtain their solutions by minimizing a pair of exterior penalty functions of their duals for a finite value of the penalty parameter θ using Newton-Armijo algorithm.

First we describe the method of converting the 1-norm TSVR as a pair of LPPs.
Let $G = [A\ e]$ be an augmented matrix. Then, by setting

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = r_1 - s_1, y-\varepsilon_1 e - G\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = p_1 - q_1, \begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = r_2 - s_2 \quad \text{and} \quad y+\varepsilon_2 e - G\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = p_2 - q_2, \tag{15}$$

where $r_1, s_1, r_2, s_2 \in R^{n+1}$ and $p_1, q_1, p_2, q_2 \in R^m$ satisfying the non-negativity constraints

$$r_1, s_1, r_2, s_2 \geq 0 \quad \text{and} \quad p_1, q_1, p_2, q_2 \geq 0,$$

the pair of QPPs (13) and (14) can be converted into the following pair of linear programming TSVR

(LPTSVR) problems:

$$\begin{aligned} \min_{r_1,s_1,\xi_1,p_1,q_1} \quad & e^t(p_1+q_1)+C_1e^t\xi_1+\eta_1e_1^t(r_1+s_1) \\ s.t. \quad & p_1-q_1+\xi_1\geq 0, \\ & G(r_1-s_1)+p_1-q_1=y-e\varepsilon_1, \\ & r_1,s_1,p_1,q_1,\xi_1\geq 0 \end{aligned} \tag{16}$$

and

$$\begin{aligned} \min_{r_2,s_2,\xi_2,p_2,q_2} \quad & e^t(p_2+q_2)+C_2e^t\xi_2+\eta_2e_1^t(r_2+s_2) \\ s.t. \quad & -p_2+q_2+\xi_2\geq 0, \\ & G(r_2-s_2)+p_2-q_2=y+e\varepsilon_2, \\ & r_2,s_2,p_2,q_2,\xi_2\geq 0, \end{aligned} \tag{17}$$

respectively, where e_1 is the vector of one's of size $(n+1)$.

Rather than solving the resulting QPPs in the dual, we first convert the constrained QPPs to unconstrained minimization problems (UMPs). Since the objective function of UMPs are not twice differentiable, both the generalization Hessian [16] and smoothing technique [25] are considered and apply fast Newton method to solve it effectively.

In a similar manner, one can easily obtain a pair of explicit LPPs for nonlinear TSVR.
Assume that the down- and up- bound nonlinear functions are expressed in the form of (8). They will be obtained by solving the following pair of 1-norm minimization problems:

$$\begin{aligned} \min_{(w_1,b_1,\xi_1)\in R^{m+1+m}} \quad & \|y-\varepsilon_1e-(K(A,A^t)w_1+b_1e)\|_1+C_1e^t\xi_1+\eta_1\|\begin{bmatrix} w_1 \\ b_1 \end{bmatrix}\|_1 \\ s.t. \quad & y-(K(A,A^t)w_1+b_1e)\geq \varepsilon_1e-\xi_1,\xi_1\geq 0 \end{aligned} \tag{18}$$

and

$$\begin{aligned} \min_{(w_2,b_2,\xi_2)\in R^{m+1+m}} \quad & \|y+\varepsilon_2e-(K(A,A^t)w_2+b_2e)\|_1+C_2e^t\xi_2+\eta_2\|\begin{bmatrix} w_2 \\ b_2 \end{bmatrix}\|_1 \\ s.t. \quad & (K(A,A^t)w_2+b_2e)-y\geq \varepsilon_2e-\xi_2,\xi_2\geq 0, \end{aligned} \tag{19}$$

where $K(A,A^t)$ is a kernel matrix.

By defining the augmented matrix $G=[K(A,A^t)\ e]$ and using (15) where $r_1,s_1,r_2,s_2\in R^{m+1}$, and proceeding as in the linear case, the pair of 1-norm minimization problems (18) and (19) can be converted into nonlinear LPTSVR problems again of the same form (16) and (17) in which e_1 becomes the vector of one's of size $(m+1)$.

Now we focus on the method of obtaining the solutions of (16) and (17) for the linear and nonlinear TSVR.

Since both the linear problems (16) and (17) are feasible and their objective functions are bounded below by zero, they are solvable. Using the optimization toolbox of MATLAB one can easily solve LPTSVR. However, because of increase in number of unknowns and constraints of LPTSVR and therefore increase in problem size, following the work of [26], we propose Newton method for LPTSVR (NLPTSVR) by considering the dual exterior penalty problems for LPTSVR as a pair of unconstrained minimization problems and solving them using Newton-Armijo method. In fact, using Proposition 1 of [26], the pair of unconstrained dual exterior penalty problems with penalty parameter $\theta>0$, corresponding to the linear

problems (16) and (17), given by

$$\min_{u,v\in R^m} L_1(u,v) = -\theta(y-\varepsilon_1 e)^t v + \frac{1}{2}\left(\left\|\begin{pmatrix} G^t v - \eta_1 e_1 \\ -G^t v - \eta_1 e_1 \\ u+v-e \\ -u-v-e \\ u - C_1 e \end{pmatrix}_+\right\|^2 + \|(-u)_+\|^2\right) \tag{20}$$

and

$$\min_{u,v\in R^m} L_2(u,v) = -\theta(y+\varepsilon_2 e)^t v + \frac{1}{2}\left(\left\|\begin{pmatrix} G^t v - \eta_2 e_1 \\ -G^t v - \eta_2 e_1 \\ -u+v-e \\ u-v-e \\ u - C_2 e \end{pmatrix}_+\right\|^2 + \|(-u)_+\|^2\right) \tag{21}$$

are solvable for all $\theta > 0$ and further there exists $\bar{\theta} > 0$ such that for any $\theta \in (0, \bar{\theta}]$ we have: for $k = 1, 2$,

$$r_k = \frac{1}{\theta}(G^t v_k - \eta_k e_1)_+, s_k = \frac{1}{\theta}(-G^t v_k - \eta_k e_1)_+, \xi_k = \frac{1}{\theta}(u_k - C_k e)_+,$$
$$p_k = \frac{1}{\theta}((-1)^{k+1}u_k + v_k - e)_+ \quad \text{and} \quad q_k = \frac{1}{\theta}((-1)^k u_k - v_k - e)_+, \tag{22}$$

where (u_1, v_1) and (u_2, v_2) are the solutions of the minimization problems (21) and (22) respectively.
In this work, we solve each of the unconstrained minimization problems by Newton-Armijo iterative algorithm. Further, using their solutions and Eqs. (15) and (22), the ε_1-insensitive down- and ε_2-insensitive up- bound functions given by (8) will be determined. Finally, the end regressor is obtained by taking their mean, i.e. by (7).

Newton Algorithm with Armijo step size for solving (20), (21) [16, 25]: For $k = 1, 2$

Choose any initial guess $\begin{bmatrix} u^0 \\ v^0 \end{bmatrix} \in R^{2m}$, where $u^0, v^0 \in R^m$.
(i) Stop the iteration if $\nabla L_k(u^i, v^i) = 0$
Else
Determine the direction vector $d_k^i \in R^{2m}$ as the solution of the following linear system of equations in $2m$ variables
$\nabla^2 L_k(u^i, v^i) d_k^i = -\nabla L_k(u^i, v^i)$
(ii) Armijo step size: Define
$\begin{bmatrix} u^{i+1} \\ v^{i+1} \end{bmatrix} = \begin{bmatrix} u^i \\ v^i \end{bmatrix} + \lambda_i d_k^i$
where $\lambda_i = \max\{1, \frac{1}{2}, \frac{1}{4}, ...\}$ is the step size in which
$L_k(u^i, v^i) - L_k((u^i, v^i) + \lambda_i d_k^i) \geq -\delta\lambda_i \nabla L_k((u^i, v^i))^t d_k^i$
and $\delta \in (0, \frac{1}{2})$.

Clearly, once the gradient vector and the Hessian matrix of $L_k(.,.)$ are known, Newton-Armijo algorithm can be performed for solving (20) and (21).

Consider the gradient of $L_k(.,.)$ which can be derived in the following form:

For $k = 1, 2$ and $u, v \in R^m$, $\nabla L_k(u,v) = \begin{bmatrix} h_{1k}(u,v) \\ h_{2k}(u,v) \end{bmatrix}_{2m\times 1}$ is such that

$$h_{1k}(u,v) = (u + (-1)^{(k+1)}v - e)_+ - (-u + (-1)^k v - e)_+ + (u - C_k e)_+ - (-u)_+$$

and

$$h_{2k}(u,v) = -\theta(y + (-1)^k \varepsilon_k e) + G[(G^t v - \eta_k e_1)_+ - (-G^t v - \eta_k e_1)_+] + ((-1)^{(k+1)}u + v - e)_+ - ((-1)^k u - v - e)_+.$$

Since the gradient function is not differentiable and therefore the Hessian matrix of second order partial derivatives of $L_k(.,.)$ does not exist in the usual sense. However, a generalized Hessian in the sense of [21] exists and is given by: For $k = 1, 2$ and $u, v \in R^m$,

$$\nabla^2 L_k(u,v) = \nabla^2 L_k = \begin{bmatrix} D_k + E_k & (-1)^{(k+1)} D_k \\ (-1)^{(k+1)} D_k & D_k + G F_k G^t \end{bmatrix}$$

where $D_k = \text{diag}(((-1)^{(k+1)}u + v - e)_*) + \text{diag}(((-1)^k u - v - e)_*), E_k = \text{diag}((u - C_k e)_*) + \text{diag}((-u)_*)$ and $F_k = \text{diag}((G^t v - \eta_k e_1)_*) + \text{diag}((-G^t v - \eta_k e_1)_*)$
in which diag(.) is a diagonal matrix.

Clearly, $\nabla^2 L_k$ is symmetric and positive semi-definite matrix of order $2m$. Since it is possible that the matrix may be ill-conditioned and therefore we will use $(\delta I + \nabla^2 L_k)^{-1}$ in place of the inverse of $\nabla^2 L_k$ where the regularization parameter δ is taken as a very small positive number.

For solving the above pair of exterior penalty problems (20) and (21), it is proposed to apply Newton-Armijo algorithm whose global convergence will follow from Proposition 4 of [26].

Lemma 3.1. *[35]: Let an invertible matrix $\bar{A}$ of the following form*

$$\bar{A} = \begin{bmatrix} \bar{P} & \bar{Q} \\ \bar{R} & \bar{S} \end{bmatrix}$$

be given where $\bar{P}, \bar{Q}, \bar{R}$ and $\bar{S}$ are block matrices of size $\bar{p} \times \bar{p}$, $\bar{p} \times \bar{s}$, $\bar{s} \times \bar{p}$ and $\bar{s} \times \bar{s}$ respectively. Assume that $\bar{A}^{-1}$can be partitioned in a similar manner

$$\bar{A}^{-1} = \begin{bmatrix} \tilde{P} & \tilde{Q} \\ \tilde{R} & \tilde{S} \end{bmatrix}$$

where $\tilde{P}, \tilde{Q}, \tilde{R}$ and $\tilde{S}$ are matrices of the same size as $\bar{P}, \bar{Q}, \bar{R}$ and $\bar{S}$ respectively. Then,
$\tilde{S} = (\bar{S} - \bar{R}\bar{P}^{-1}\bar{Q})^{-1}$, $\tilde{P} = \bar{P}^{-1} + \bar{P}^{-1}\bar{Q}\tilde{S}\bar{R}\bar{P}^{-1}$, $\tilde{Q} = -\bar{P}^{-1}\bar{Q}\tilde{S}$ *and* $\tilde{R} = -\tilde{S}\bar{R}\bar{P}^{-1}$.

Theorem 3.2. *For $k = 1, 2$ and $u, v \in R^m$, let the inverse of the matrix $(\delta I + \nabla^2 L_k)(u,v)$ be taken to be of the following form:*

$$(\delta I + \nabla^2 L_k)^{-1} = \begin{bmatrix} \delta I + D_k + E_k & (-1)^{(k+1)} D_k \\ (-1)^{(k+1)} D_k & \delta I + D_k + G F_k G^t \end{bmatrix}^{-1} = \begin{bmatrix} \tilde{P}_k & \tilde{Q}_k \\ \tilde{R}_k & \tilde{S}_k \end{bmatrix}.$$

Then, we can write
$\tilde{P}_k = M_k + M_k D_k \tilde{S}_k D_k M_k$, $\tilde{Q}_k = (-1)^k D_k M_k \tilde{S}_k$, $\tilde{R}_k = (-1)^k \tilde{S}_k D_k M_k$ *and*

$$\tilde{S}_k = [\delta I + (\delta I + E_k) M_k D_k + G F_k G^t]^{-1},$$

where $M_k = (\delta I + D_k + E_k)^{-1}$ is a diagonal matrix of order m.

Proof. Consider the expression $(\delta I + D_k + G F_k G^t) - D_k M_k D_k$

$$= (\delta I + G F_k G^t) + M_k^{-1} M_k D_k - D_k M_k D_k$$

$$= (\delta I + G F_k G^t) + (M_k^{-1} - D_k) M_k D_k = (\delta I + (\delta I + E_k) M_k D_k + G F_k G^t).$$

Now, using Lemma 3.1, it is easy to verify the result. □

Note that since $(\delta I + D_k + E_k)$ is positive definite, M_k exists and is also a diagonal matrix of order m.

For $k = 1, 2$ and $u, v \in R^m$, consider the following system of linear equations in augmented variables $[z_{1k}$ $z_{2k}]$:

$$(\delta I + (\delta I + E_k)M_k D_k + GF_k G^t)[z_{1k} \quad z_{2k}] = [D_k M_k h_{1k}(u,v) \quad h_{2k}(u,v)], \tag{23}$$

where $z_{1k} = z_{1k}(u,v), z_{2k} = z_{2k}(u,v) \in R^m$. Since the solution of the above system can be written as

$$z_{1k} = \tilde{S}_k D_k M_k h_{1k}(u,v) \quad \text{and} \quad z_{2k} = \tilde{S}_k h_{2k}(u,v), \tag{24}$$

the direction vector $d_k^i \in R^{2m}$, for *i=0, 1, 2, ...*, of Newton-Armijo algorithm can be computed as:

$$d_k^i = -(\delta I + \nabla^2 L_k)^{-1} \nabla L_k(u^i, v^i) = -\begin{bmatrix} \tilde{P}_k & \tilde{Q}_k \\ \tilde{R}_k & \tilde{S}_k \end{bmatrix} \begin{bmatrix} h_{1k}(u^i, v^i) \\ h_{2k}(u^i, v^i) \end{bmatrix}$$

$$= -\begin{bmatrix} M_k h_{1k}(u^i, v^i) + M_k D_k (z_{1k} + (-1)^k z_{2k})(u^i, v^i) \\ (-1)^k (z_{1k} + (-1)^k z_{2k})(u^i, v^i) \end{bmatrix}.$$

From the above derivations one can observe that the direction vector d_k^i at the i-th iteration of the algorithm can be determined by solving a matrix equation of order m for unknown augmented variables of size $m \times 2$ rather than solving the original matrix equation of order *2m.*

For the linear case where suppose $n << m$, it is preferable to use SMW identity [19] to calculate the direction vector d_k^i at the i-th iteration of the algorithm since it has the advantage that a linear system of equations of size $(n + 1) \times (n + 1)$ needs to be solved rather than solving a linear system of equations of larger size $(m \times m)$. In fact, by defining $H_k = GF_k^{1/2} = [A \; e]F_k^{1/2}$, $L_k = (\delta I + (\delta I + E_k)M_k D_k)$ and applying the SMW identity [19] we have:

$$\tilde{S}_k = (L_k + H_k H_k^t)^{-1} = L_k^{-1} - L_k^{-1} H_k (I + H_k^t L_k^{-1} H_k)^{-1} H_k^t L_k^{-1}.$$

Since L_k is a diagonal matrix, its inverse can be trivially computed.

In this work, we solve both the problems (20) and (21) using Newton's method, i.e. the unknown (u^{i+1}, v^{i+1}) at the (i+1)-th iteration is obtained by solving the system of equations

$$\nabla L_k(u^i, v^i) + (\delta I + \nabla^2 L_k(u^i, v^i))((u^{i+1}, v^{i+1}) - (u^i, v^i)) = 0, \tag{25}$$

where $k = 1, 2$ and *i=0, 1, 2, ...*

4. Numerical Experiments and Comparison of Results

In addition to the Newton method of solving LPTSVR problem using generalized Hessian discussed in the previous section, we also solved the proposed LPTSVR numerically by smoothing technique, i.e. replacing the plus function appearing in the objective functions of the unconstrained dual exterior penalty problems given by (20) and (21) with a smooth approximation function and solving a pair of modified problems by Newton method. In fact, we replaced the non-differential plus function x_+ by the following smooth function [25]:

$$p(x, \alpha) = x + \frac{1}{\alpha} \log(1 + \exp(-\alpha x)),$$

where $\alpha > 0$ is the smoothing parameter.

All the regression methods were implemented in MATLAB R2010a environment on a PC running on Windows XP OS with 2.27 GHz Intel(R) Xeon(R) processor having 3 GB of RAM. The standard SVR was solved by LIBSVM [7] and for the implementation of TSVR and LPTSVR, we used the optimization toolbox of MATLAB.

To demonstrate the effectiveness of the proposed Newton LPTSVR, computational testing was carried-out in this section on seven synthetic and several well-known, publicly available, real-world benchmark

datasets and their results were compared with smooth LPTSVR (SLPTSVR), LPTSVR, TSVR and SVR. In all the examples considered, the Gaussian nonlinear kernel function with parameter $\mu > 0$, defined by: for $x_1, x_2 \in R^m$

$$k(x_1, x_2) = \exp(-\mu||x_1 - x_2||^2),$$

is used. The 2-norm root mean square error (RMSE) test error

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2}$$

and standard deviation are adopted to measure the performance of each algorithm.

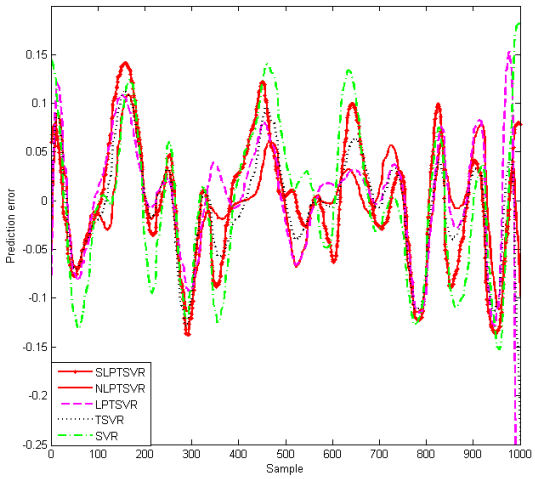
Since more parameters need to be selected in TSVR, LPTSVR, SLPTSVR and NLPTSVR, this will lead to, however, a slower model selection speed in comparison to SVR and therefore in all experiments $\varepsilon_1 = \varepsilon_2 = 0.01$ and $C_1 = C_2$ were assumed. Further we set $\eta_1 = \eta_2 = 10^{-5}$ and the penalty parameter $\theta = 0.1$. For SVR, the insensitive margin parameter ε is taken as 0.01 and for SLPTSVR the smoothing parameter $\alpha = 5$. The optimal values of the parameters were determined by performing 10-fold cross-validation on the training dataset. In detail, each data is partitioned into ten subsets with similar sizes and distributions. Then, the union of nine subsets is used as the training set while the remaining subset is used as the test. The experiment is repeated ten times such that every subset is used once as a test set. The whole process described above is then repeated ten times with randomly generated cost matrices belonging to the same cost type, and the average results are recorded as the final results, where statistical significance are examined. The values of the regularization parameter $C_1 = C_2 = C$ were allowed to vary from the set $\{10^{-5}, 10^{-4}, ..., 10^5\}$. However, the kernel parameter μ assumed values from $\{2^{-5}, 2^{-4}, ..., 2^2\}$ for LPTSVR and for the rest of the methods from $\{2^{-10}, 2^{-9}, ..., 2^{10}\}$. Finally, choosing these optimal values, the RMSE on the test dataset was calculated.

4.1. Synthetic datasets

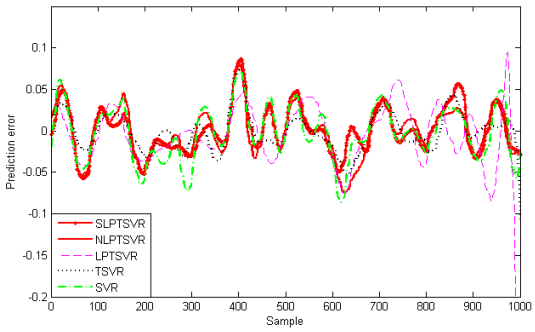
To demonstrate the performance of the proposed method, we first considered the following function defined as [36]:

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

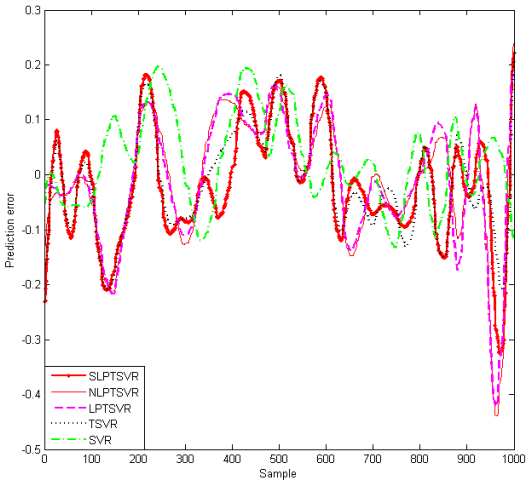
Training samples were polluted by adding three kinds of different noises: (i) uniformly distributed noises over the interval $[-0.2, 0.2]$; (ii) Gaussian noises with mean zero and standard deviation 0.05; (iii) Gaussian noises with mean zero and standard deviation 0.2, i.e. we have taken the training samples (x_i, y_i) of the form $y_i = f(x_i) + \varsigma_i$, where ς_i is the noise. 200 training and 1000 test samples were generated randomly under uniform distribution over the interval $[0, 6]$. Test data was assumed to be noise-free. The estimation functions obtained with one run simulation results for these three different types of noises were illustrated in Figure 1a, Figure 1b and Figure 1c. The noisy samples were represented by the symbol " $\circ$ ". The numerical results along with their computational learning time in seconds obtained by NLPTSVR, SLPTSVR, LPTSVR, TSVR and SVR were summarized in Table 2.



a) Uniformly distributed noises over[-0.2,0.2].



b) Gaussian noises with mean zero and standard deviation 0.05.



c) Gaussian noises with mean zero and standard deviation 0.2.

Figure 1: Prediction error over the test set by SLPTSVR, NLPTSVR, LPTSVR, SVR and TSVR for the simulated dataset generated by the function $sin(x)cos(x^2)$ where three different kinds of additive noises were used. Gaussian kernel was employed.

To further investigate the effectiveness of NLPTSVR and SLPTSVR another six synthetic datasets, each consisting of 200 training and 1000 test samples, were generated using functions whose definitions are given in Table 1. As in the first example discussed above, the observed values of the training set were polluted by additive noises. For each kind of noises, ten independent groups of samples were randomly generated. The average accuracies of NLPTSVR, SLPTSVR, LPTSVR, TSVR and SVR with 10 independent runs along with their computational training time in seconds were shown in Table 2. Clearly one can observe from Table 2 that, in comparison to SVR and TSVR, both NLPTSVR and SLPTSVR show either better or comparable generalization performance with fast learning speed.

Table 1: Functions used for generating synthetic datasets.

Name	Function Definition	Domain of Definition
Function 1	$f(x) = sin(x)cos(x^2)$	$x \in [0, 6]$
Function 2	$f(x_1, x_2) = \frac{sin\sqrt{(x_1^2+x_2^2)}}{\sqrt{(x_1^2+x_2^2)}}$	$x_1, x_2 \in [-4\pi, 4\pi]$
Function 3	$f(x_1, x_2, x_3, x_4, x_5) = 10sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5$	$x_1, x_2, x_3, x_4, x_5 \in [0, 1]$
Function 4	$f(x_1, x_2) = \frac{(5-x_2)^2}{3(5-x_1)^2+(5-x_2)^2}$	$x_1, x_2 \in [0, 10]$
Function 5	$f(x_1, x_2) = exp(x_1 sin(\pi x_2))$	$x_1, x_2 \in [-1, 1]$
Function 6	$f(x_1, x_2) = 1.9\Big[1.35 + exp(x_1)sin(13(x_1 - 0.6)^2)$ $+exp(3(x_2 - 0.5))sin(4\pi(x_2 - 0.9)^2)\Big]$	$x_1, x_2 \in [0, 1]$
Function 7	$f(x) = sin\left(\frac{2\pi(0.35\times10+1)}{0.35x+1}\right)$	$x \in [0, 10]$

Table 2: Performance comparison of NLPTSVR with SLPTSVR, SVR, TSVR and LPTSVR on ten independent runs. RMSE was used for comparison. Gaussian kernel was employed.

Dataset	Training set size	Test set size	SVR Time(s)	TSVR Time(s)	LPTSVR Time(s)	SLPTSVR Time(s)	NLPTSVR Time(s)
Function1	200x1	1000x1	0.1044	0.0453	0.0354	0.0614	0.0456
			0.3675	0.2971	0.3154	0.0210	0.0159
Function2	200x2	1000x2	0.2013	0.1883	0.1578	0.1250	0.1322
			0.6714	0.1845	0.1689	0.0310	0.0165
Function3	200x5	1000x5	0.2022	0.2188	0.0454	0.0326	0.0242
			0.5034	0.3134	0.4568	0.0598	0.0230
Function4	200x2	1000x2	0.1237	0.1393	0.1552	0.1360	0.1395
			0.3043	0.3522	0.2976	0.0870	0.0939
Function5	200x2	1000x2	0.0224	0.0283	0.0255	0.0365	0.0122
			0.4864	0.3744	0.3522	0.0577	0.0765
Function6	200x2	1000x2	0.0976	0.0235	0.0278	0.0268	0.0361
			0.2025	0.1337	0.1566	0.1020	0.0230
Function7	200x1	1000x1	0.1355	0.1055	0.0822	0.0742	0.0761
			0.2765	0.1199	0.1230	0.0520	0.0220

a) Uniformly distributed noises over [-0.2,0.2]

Dataset	Training set size	Test set size	SVR Time(s)	TSVR Time(s)	LPTSVR Time(s)	SLPTSVR Time(s)	NLPTSVR Time(s)
Function1	200x1	1000x1	0.0964	0.0227	0.0296	0.0293	0.0369
			0.6453	0.5935	0.4087	0.0532	0.0459
Function2	200x2	1000x2	0.0576	0.0437	0.0311	0.0350	0.0163
			0.2336	0.1350	0.1076	0.0206	0.0190
Function3	200x5	1000x5	0.1652	0.1842	0.0497	0.0308	0.0555
			0.2675	0.1655	0.1234	0.0930	0.0117
Function4	200x2	1000x2	0.0655	0.0773	0.0120	0.0158	0.0071
			0.2034	0.1629	0.2025	0.0376	0.0502
Function5	200x2	1000x2	0.0120	0.0143	0.0095	0.0062	0.0132
			0.1804	0.1402	0.1577	0.0766	0.0802
Function6	200x2	1000x2	0.0961	0.0154	0.1173	0.1075	0.0728
			0.3098	0.1322	0.1066	0.0211	0.0313
Function7	200x1	1000x1	0.0875	0.0560	0.0222	0.0178	0.0322
			0.5673	0.1439	0.3777	0.1098	0.0355

b) Gaussian noises with mean zero and standard deviation 0.05

Dataset	Training set size	Test set size	SVR Time(s)	TSVR Time(s)	LPTSVR Time(s)	SLPTSVR Time(s)	NLPTSVR Time(s)
Function1	200x1	1000x1	0.1158	0.0885	0.1286	0.1185	0.0740
			0.4555	0.4577	0.3562	0.0657	0.0762
Function2	200x2	1000x2	0.0480	0.1558	0.0974	0.0723	0.0853
			0.5760	0.7367	0.6097	0.0764	0.0543
Function3	200x5	1000x5	0.2252	0.2089	0.1633	0.0820	0.1215
			0.3230	0.1612	0.2085	0.0222	0.0106
Function4	200x2	1000x2	0.1102	0.1245	0.1098	0.0979	0.1176
			0.5430	0.1736	0.4087	0.0480	0.529
Function5	200x2	1000x2	0.0634	0.0466	0.0884	0.0418	0.0233
			0.2453	0.2025	0.2103	0.0290	0.0176
Function6	200x2	1000x2	0.0497	0.0512	0.0752	0.0623	0.0495
			0.3980	0.0818	0.1023	0.0340	0.0154
Function7	200x1	1000x1	0.2262	0.2057	0.1176	0.1082	0.0975
			0.4543	0.3998	0.2307	0.0439	0.0210

c) Gaussian noises with mean zero and standard deviation 0.2

4.2. Real-world benchmark datasets

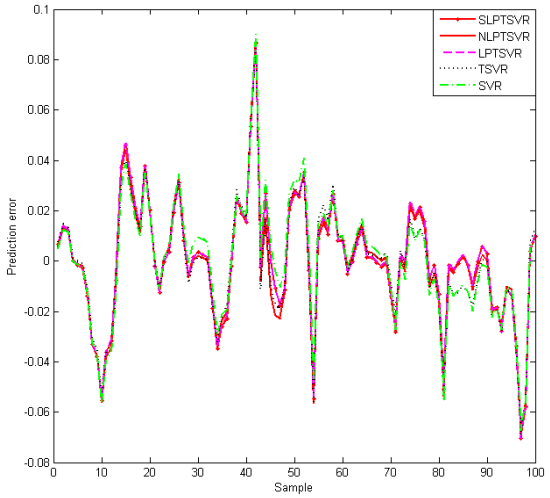
In all the real-world examples considered, we normalized the given data in the following manner:

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

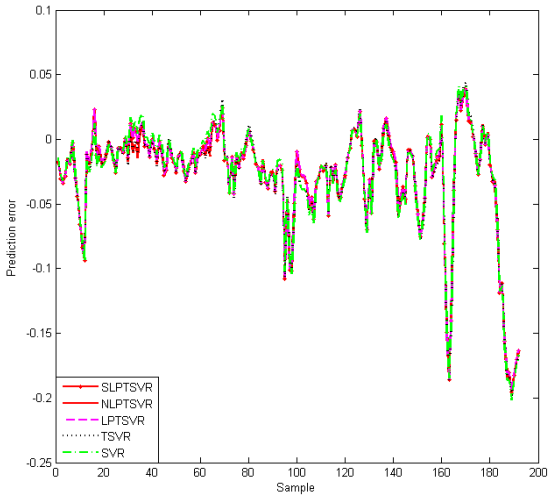
where $x_j^{\min} = \min_{i=1}^{m}(x_{ij})$ and $x_j^{\max} = \max_{i=1}^{m}(x_{ij})$ denote the minimum and maximum values, respectively, of the j-th column of A, $x_{ij}$is the (i,j)-th element of the input matrix A and $\bar{x}_{ij}$ is its corresponding normalized value.

As the first real-world dataset, we considered the example of Box and Jenkins gas furnace [5]. 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 2 attributes taken

to be of the form [36]: $x(t) = (y(t-1), u(t-4))$. Thus we get 292 samples in total where each sample is of the form$(x(t), y(t))$. The first 100 samples were chosen for training and the rest for testing. The performances of SVR, TSVR, LPTSVR, SLPTSVR and NLPTSVR on the training and test sets were shown in Figure 2a and Figure 2b respectively.



a) Prediction over the Training set



b) Prediction over the Test set

Figure 2: Prediction error over the whole dataset by SLPTSVR, NLPTSVR, LPTSVR, SVR and TSVR for the gas furnace dataset. Gaussian kernel was employed.

To further demonstrate the validity of NLPTSVR, we tested nonlinearly on several benchmark datasets including: Wine quality white, Abalone, Concrete CS, Boston, Forest fires, Flares, Machine CPU, Auto-MPG original and Servo from UCI repository [31]; Kin-fh, Bank-32fh and Demo from DELVE [13]; the time series datasets generated by the Mackey-Glass differential equation, SantaFeA and Sunspots

taken from the web site: http://www.cse.ogi.edu/~ericwan; Bodyfat and NO2 from StatLib collection: http://lib.stat.cmu.edu/datasets and a number of interesting financial datasets of stock index taken from the web site: http://www.dailyfinance.com. In addition two more time series datasets generated by the Lorenz differential equation were also considered.

Consider the Mackey-Glass time delay differential equation [29] defined by:

$$\frac{dx(t)}{dt} = -0.1x(t) + \frac{0.2x(t-\tau)}{1+x(t-\tau)^{10}},$$

for the parameter values *a=0.2, b=0.1* and $\tau = 17, 30$. Let the time series generated by the above equation corresponding to $\tau = 17$ and $\tau = 30$ be denoted by MG_{17} and MG_{30} respectively. Assume that five previous values were used to predict the current value. Among the total of 1495 samples obtained, the first 500 were considered for training and the rest for testing.

As examples of financial datasets, the stock index of S&P500, Google, IBM, Intel, Redhat, and Microsoft were taken. For our experimental study we considered 755 closing prices starting from 01-01-2006 to 31-12-2008 and using five previous values to predict the current value, 750 samples in total were obtained. For each of the above financial time series examples considered except S&P500, the first 200 samples were taken for training and the rest 550 for testing. For S&P500 we choose, however, the first 300 samples for training and the rest for testing.

By assuming different sampling rates $\tau = 0.05$ and $\tau = 0.20$ with parameters values $\rho = 10$, $r = 28$ and $b = 8/3$, two time series datasets $\text{Lorenz}_{0.05}$ and $\text{Lorenz}_{0.20}$ are generated using the time series values associated to the variable x of the Lorenz differential equation [29] given by:

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

obtained by fourth-order Runge-Kutta method. They consist of 30000 number of time series values. To avoid the initial transients the first 1000 values were discarded. The next 3000 values were taken for our experiment and using five previous values to predict the current value, 2995 samples in total were obtained. Among them, the first 500 samples were taken for training and the remaining 2495 samples for testing.

Table 3: Performance comparison of NLPTSVR with SLPTSVR, SVR, TSVR and LPTSVR on real world datasets. RMSE was used for comparison. Gaussian kernel was employed.

Datasets (Train Size,Test Size)	SVR RMSE±STD Time(s) (C, μ)	TSVR RMSE±STD Time(s) $(C_1 = C_2, \mu)$	LPTSVR RMSE±STD Time(s) $(C_1 = C_2, \mu)$	SLPTSVR RMSE±STD Time(s) $(C_1 = C_2, \mu)$	NLPTSVR RMSE±STD Time(s) $(C_1 = C_2, \mu)$
Gas furnace	0.0518 ± 0.2402	0.0517 ± 0.2335	0.0509 ± 0.2231	0.0510 ± 0.2229	0.0512 ± 0.2228
$(100 \times 2, 192 \times 2)$	0.1045	0.0832	0.0888	0.0120	0.0170
	$(10^0, 2^{-3})$	$(10^{-1}, 2^{-3})$	$(10^{-3}, 2^{-3})$	$(10^{-2}, 2^{-3})$	$(10^{-4}, 2^{-4})$
Wine quality-white	0.1715 ± 0.1065	0.2029 ± 0.1455	0.2290 ± 0.0826	0.1360 ± 0.0714	0.1689 ± 0.0603
$(1000 \times 11, 3898 \times 11)$	4.5238	2.568	1.2387	0.9740	0.5620
	$(10^2, 2^5)$	$(10^{-1}, 2^{-2})$	$(10^0, 2^{-3})$	$(10^{-8}, 2^{-5})$	$(10^0, 2^{-1})$
Abalone	0.1901 ± 0.1892	0.1188 ± 0.1185	0.1341 ± 0.1296	0.1126± 0.1209	0.1206 ± 0.1304
$(1000 \times 8, 3177 \times 8)$	13.6482	2.7266	4.2390	0.6532	0.7251
	$(10^2, 2^{-4})$	$(10^{-1}, 2^{-3})$	$(10^0, 2^{-4})$	$(10^0, 2^{-1})$	$(10^0, 2^{-1})$
Concrete CS	0.1526± 0.2031	0.1466± 0.1930	0.1517 ± 0.1681	0.1447 ± 0.1560	0.1435± 0.1553
$(700 \times 8, 330 \times 8)$	6.2765	2.802	2.0372	0.8722	0.7736
	$(10^1, 2^0)$	$(10^0, 2^{-4})$	$(10^1, 2^{-5})$	$(10^{-1}, 2^{-1})$	$(10^{-1}, 2^{-1})$
Boston	0.1237 ± 0.1782	0.1393 ± 0.2202	0.1406 ± 0.2241	0.1280 ± 0.1796	0.1292 ± 0.1822

Continued on next page

Table 3 – *Continued from previous page*

Datasets (Train Size,Test Size)	SVR RMSE±STD Time(s) (C, μ)	TSVR RMSE±STD Time(s) $(C_1 = C_2, \mu)$	LPTSVR RMSE±STD Time(s) $(C_1 = C_2, \mu)$	SLPTSVR RMSE±STD Time(s) $(C_1 = C_2, \mu)$	NLPTSVR RMSE±STD Time(s) $(C_1 = C_2, \mu)$
$(200 \times 13, 306 \times 13)$	0.2786	0.0836	0.0763	0.0204	0.0127
	$(10^1, 2^{-3})$	$(10^{-2}, 2^0)$	$(10^0, 2^{-3})$	$(10^0, 2^{-5})$	$(10^0, 2^{-5})$
Forest fires	0.1569 ± 0.1076	0.0765 ± 0.0735	0.0863 ± 0.0423	0.0756 ± 0.0045	0.0755 ± 0.0026
$(200 \times 12, 318 \times 12)$	0.3204	0.1378	0.1232	0.0755	0.0891
	$(10^3, 2^{-8})$	$(10^{-1}, 2^{-8})$	$(10^0, 2^{-5})$	$(10^0, 2^{-4})$	$(10^0, 2^{-4})$
Flares	0.1896 ± 0.1347	0.1274 ± 0.1164	0.1324 ± 0.0767	0.1266 ± 0.0651	0.1594 ± 0.1222
$(200 \times 9, 866 \times 9)$	0.2036	0.1218	0.1003	0.0334	0.0210
	$(10^3, 2^{-8})$	$(10^{-1}, 2^{-8})$	$(10^0, 2^{-5})$	$(10^0, 2^{-4})$	$(10^0, 2^{-4})$
Machine CPU	0.0443 ± 0.3287	0.0377 ± 0.2081	0.0232 ± 0.2055	0.0334 ± 0.1893	0.0284 ± 0.1983
$(150 \times 7, 59 \times 7)$	0.2265	0.1230	0.1325	0.0939	0.0231
	$(10^5, 2^{-8})$	$(10^{-2}, 2^{-2})$	$(10^0, 2^{-3})$	$(10^5, 2^{-2})$	$(10^0, 2^{-2})$
Autp-MPG Original	0.1490 ± 0.2056	0.2183 ± 0.2146	0.1665 ± 0.1949	0.1552 ± 0.1830	0.1551 ± 0.1832
$(100 \times 7, 292 \times 7)$	0.0766	0.0181	0.0137	0.0090	0.0104
	$(10^2, 2^{-4})$	$(10^{-5}, 2^{10})$	$(10^0, 2^{-2})$	$(10^{-1}, 2^{-1})$	$(10^{-2}, 2^{-1})$
Servo	0.1240 ± 0.2130	0.1330 ± 0.2197	0.1277 ± 0.2547	0.1321± 0.2337	0.1322 ± 0.2333
$(100 \times 4, 67 \times 4)$	0.0972	0.0598	0.0487	0.0120	0.0120
	$(10^5, 2^{-3})$	$(10^{-1}, 2^0)$	$(10^{-4}, 2^0)$	$(10^5, 2^1)$	$(10^5, 2^1)$
Kin-fh	0.0947 ± 0.1373	0.0952 ± 0.1432	0.0991 ± 0.1321	0.0927 ± 0.1327	0.0929 ± 0.1320
$(1000 \times 32, 7192 \times 32)$	15.2765	4.2083	2.9762	0.0972	0.1076
	$(10^{-1}, 2^{-4})$	$(10^{-5}, 2^{-7})$	$(10^{-3}, 2^{-2})$	$(10^1, 2^{-6})$	$(10^1, 2^{-6})$
Bank-32fh	0.1497 ± 0.2006	0.1431 ± 0.1989	0.1636 ± 0.1308	0.1418 ± 0.1064	0.1425 ± 0.1003
$(1000 \times 21, 700 \times 21)$	5.6217	5.1196	2.3154	1.1273	1.2008
	$(10^3, 2^{-10})$	$(10^0, 2^{-8})$	$(10^5, 2^{-1})$	$(10^1, 2^{-5})$	$(10^1, 2^{-6})$
Demo	0.1117 ± 0.0876	0.1046 ± 0.1088	0.1006 ± 0.0628	0.1028 ± 0.0833	0.1030 ± 0.0839
$(500 \times 4, 1548 \times 4)$	1.3028	0.7454	0.6530	0.0865	0.0622
	$(10^0, 2^3)$	$(10^{-1}, 2^1)$	$(10^0, 2^0)$	$(10^0, 2^3)$	$(10^0, 2^3)$
MG_{17}	0.0084 ± 0.2670	0.0051 ± 0.2540	0.0047 ± 0.2539	0.0061 ± 0.2537	0.0063 ± 0.2538
$(500 \times 5, 995 \times 5)$	2.8720	0.8362	0.7523	0.0420	0.0555
	$(10^4, 2^2)$	$(10^3, 2^3)$	$(10^4, 2^2)$	$(10^1, 2^3)$	$(10^0, 2^3)$
MG_{30}	0.0275 ± 0.2586	0.0218 ± 0.2495	0.0265 ± 0.2480	0.0247 ± 0.2478	0.0247 ± 0.2479
$(500 \times 5, 995 \times 5)$	2.1430	0.5100	0.4823	0.0492	0.0230
	$(10^2, 2^1)$	$(10^{-2}, 2^3)$	$(10^3, 2^2)$	$(10^1, 2^3)$	$(10^1, 2^3)$
SantafeA	0.0423 ± 0.1702	0.0449 ± 0.1775	0.0446 ± 0.1938	0.0449 ± 0.0449	0.0451 ± 0.1884
$(200 \times 5, 795 \times 5)$	0.3208	0.1485	0.1076	0.0108	0.0245
	$(10^1, 2^2)$	$(10^{-1}, 2^1)$	$(10^0, 2^1)$	$(10^{-3}, 2^3)$	$(10^{-1}, 2^3)$
Sunspots	0.0845 ± 0.2301	0.0816 ± 0.2249	0.0831 ± 0.2111	0.0798 ± 0.2114	0.0794 ± 0.2114
$(100 \times 5, 190 \times 5)$	0.0923	0.0785	0.0544	0.0092	0.0133
	$(10^2, 2^{-3})$	$(10^{-1}, 2^{-2})$	$(10^0, 2^{-3})$	$(10^1, 2^1)$	$(10^1, 2^1)$
Body fat	0.0126 ± 0.1660	0.0151 ± 0.1664	0.0212 ± 0.1693	0.0196 ± 0.1632	0.0222 ± 0.1607
$(150 \times 14, 102 \times 14)$	0.3109	0.2649	0.2230	0.0692	0.0633
	$(10^3, 2^{-9})$	$(10^{-1}, 2^{-6})$	$(10^0, 2^{-5})$	$(10^0, 2^{-4})$	$(10^0, 2^{-4})$
NO2	0.1456 ± 0.1472	0.1350 ± 0.1470	0.1491 ± 0.1473	0.1454 ± 0.1317	0.1442 ± 0.1340
$(100 \times 7, 400 \times 7)$	0.1253	0.0481	0.0372	0.0222	0.0117
	$(10^5, 2^{-9})$	$(10^{-1}, 2^{-8})$	$(10^{-2}, 2^{-5})$	$(10^0, 2^{-4})$	$(10^0, 2^{-4})$
S&P500	0.0384 ± 0.2510	0.0497 ± 0.2445	0.0280 ± 0.2414	0.0286 ± 0.2420	0.0306 ± 0.2413

Continued on next page

Table 3 – *Continued from previous page*

Datasets (Train Size,Test Size)	SVR RMSE±STD Time(s) (C, μ)	TSVR RMSE±STD Time(s) $(C_1 = C_2, \mu)$	LPTSVR RMSE±STD Time(s) $(C_1 = C_2, \mu)$	SLPTSVR RMSE±STD Time(s) $(C_1 = C_2, \mu)$	NLPTSVR RMSE±STD Time(s) $(C_1 = C_2, \mu)$
$(300 \times 5, 450 \times 5)$	0.1732	0.1485	0.1220	0.0440	0.0521
	$(10^3, 2^{-3})$	$(10^{-1}, 2^{-4})$	$(10^0, 2^{-5})$	$(10^0, 2^1)$	$(10^{-5}, 2^1)$
Google	0.0252 ± 0.1872	0.0265 ± 0.1900	0.0260 ± 0.1878	0.0288 ± 0.1829	0.0329 ± 0.1798
$(200 \times 5, 550 \times 5)$	0.3023	0.1334	0.1133	0.0321	0.0301
	$(10^3, 2^{-8})$	$(10^{-1}, 2^{-5})$	$(10^0, 2^{-5})$	$(10^0, 2^{-3})$	$(10^{-1}, 2^{-3})$
IBM	0.0318 ± 0.2268	0.0729 ± 0.2238	0.0318 ± 0.2272	0.0425 ± 0.2392	0.0333 ± 0.2262
$(200 \times 5, 550 \times 5)$	0.1420	0.1892	0.1674	0.0267	0.0433
	$(10^3, 2^{-7})$	$(10^5, 2^{-4})$	$(10^{-1}, 2^{-5})$	$(10^5, 2^{-3})$	$(10^0, 2^{-3})$
Intel	0.0340 ± 0.2102	0.0369 ± 0.2104	0.0456 ± 0.2168	0.0424 ± 0.2129	0.0369 ± 0.2060
$(200 \times 5, 550 \times 5)$	0.1920	0.1416	0.1002	0.0108	0.0087
	$(10^3, 2^{-7})$	$(10^{-1}, 2^{-3})$	$(10^0, 2^{-3})$	$(10^{-1}, 2^{-3})$	$(10^0, 2^{-3})$
Red Hat	0.0334 ± 0.1823	0.0367 ± 0.2070	0.0342 ± 0.2006	0.0355 ± 0.1982	0.0361 ± 0.1973
$(200 \times 5, 550 \times 5)$	0.4021	0.2131	0.1562	0.0350	0.0344
	$(10^3, 2^{-9})$	$(10^0, 2^{-5})$	$(10^0, 2^{-5})$	$(10^{-1}, 2^{-3})$	$(10^{-3}, 2^{-3})$
Microsoft	0.0319 ± 0.1823	0.0313 ± 0.1796	0.0310 ± 0.1789	0.0315 ± 0.1786	0.0315 ± 0.1788
$(200 \times 5, 550 \times 5)$	0.2354	0.1141	0.0988	0.0113	0.0086
	$(10^4, 2^{-10})$	$(10^{-1}, 2^{-5})$	$(10^1, 2^{-5})$	$(10^1, 2^{-3})$	$(10^1, 2^{-3})$
$\text{Lorenz}_{0.05}$	0.0067 ± 0.2270	0.0035 ± 0.2290	0.0035 ± 0.2281	0.0035 ± 0.2282	0.0035 ± 0.2283
$(500 \times 5, 2495 \times 5)$	2.9212	0.6432	0.4827	0.1022	0.0922
	$(10^4, 2^{-10})$	$(10^{-1}, 2^0)$	$(10^4, 2^{-1})$	$(10^5, 2^0)$	$(10^5, 2^0)$
$\text{Lorenz}_{0.2}$	0.0049 ± 0.1902	0.0050 ± 0.2054	0.005 ± 0.2041	0.0054 ± 0.2037	0.005 ± 0.2042
$(500 \times 5, 2495 \times 5)$	2.8613	0.5302	0.6103	0.0822	0.0915
	$(10^4, 2^{-6})$	$(10^{-1}, 2^0)$	$(10^0, 2^{-2})$	$(10^5, 2^1)$	$(10^5, 2^0)$

Table 4: Average ranks of SVR, TSVR, LPTSVR, SLPTSVR and NLPTSVR with Gaussian kernel on RMSE values.

Datasets	SVR	TSVR	LPTSVR	SLPTSVR	NLPTSVR
Gas furnace	5	4	1	2	3
Wine quality-white	3	4	5	1	2
Abalone	5	2	4	1	3
Concrete CS	5	3	4	2	1
Boston	1	4	5	2	3
Forest fires	5	4	3	2	1
Flares	5	2	3	1	4
Machine CPU	5	4	1	3	2
Auto-MPG original	1	5	4	3	2
Servo	1	5	2	3	4
Kin-fh	3	4	5	1	2
Bank-32fh	4	3	5	1	2
Demo	5	4	1	2	3
MG17	5	2	1	3	4
MG30	5	1	4	2.5	2.5
SantafeA	1	3.5	2	3.5	5
Sunspots	5	3	4	2	1
Body fat	1	2	4	3	5
NO2	4	1	5	3	2
S&P500	4	5	1	2	3
Google	1	3	2	4	5
IBM	1.5	5	1.5	4	3
Intel	1	2.5	5	4	2.5
Red Hat	1	5	2	3	4
Microsoft	5	2	1	3.5	3.5
Lorenz0.05	5	2.5	2.5	2.5	2.5
Lorenz0.2	1	2.8	2.8	2.8	2.8
Average Rank	3.2777	3.2703	2.9925	2.4740	2.8814

For all the datasets considered the number of training and test samples chosen, the number of attributes, the optimal parameter values determined using 10-fold cross-validation and the numerical results along with their computational training time in seconds obtained by NLPTSVR, SLPTSVR, LPTSVR, TSVR and SVR were summarized in Table 3. From the Table 3, we notice that generalization capability of our proposed algorithms NLPTSVR and SLPTSVR are better than SVR and TSVR on many of the datasets considered. Specifically, our proposed algorithm NLPTSVR gains comparable or better accuracy with TSVR on 18 of 27 datasets and SLPTSVR gains comparable or better accuracy on 19 of 27 datasets. In addition, NLPTSVR and SLPTSVR achieve its best performance on Wine quality-white, Abalone, Concrete CS, Forest fires, Flares, Kin-fh, Bank-32fh, Sunspots and $\text{Lorenz}_{0.2}$. Furthermore, the training time of our proposed methods greatly outperformed SVR and TSVR on all the benchmark datasets considered. The average ranks of all the algorithms on RMSE for Gaussian kernel were computed and listed in Table 4. One can observed from Table 4 that our proposed algorithms LPTSVR, SLPTSVR and NLPTSVR outperform SVR and TSVR.

5. Conclusions and Future Works

We have presented 1-norm twin support vector regression as a pair of linear programming problems whose solutions were obtained by solving their exterior penalty dual problems as a pair of unconstrained minimization problems using Newton-Armijo algorithm. Though the proposed method requires the solution of a pair of unconstrained minimization problems having $2m$ variables and hence a pair of linear

equation solvers of size $2m$ at each iteration of the algorithm, where m is the number of training examples, using the properties of block matrices it was shown that the solution can be obtained using a pair of linear equation solvers of size m for unknown augmented variables of size $m \times 2$. The method was implemented in MATALB without using commands of optimization toolbox. Numerical results show that the proposed method is a very promising computational tool for regression problems. It should be pointed out that there are four parameters in our proposed formulation and parameters selection is a practical problem and should be addressed in future study.

References

[1] S Balasundaram, D Gupta, Training Lagrangian twin support vector regression via unconstrained convex minimization, Knowledge-Based Systems 59 (2014) 85-96.
[2] S Balasundaram, D Gupta, On implicit Lagrangian twin support vector regression by Newton method, International Journal of Computational Intelligence Systems 7 (1) (2014) 50-64.
[3] S Balasundaram, M Tanveer, On Lagrangian twin support vector regression. Neural Computing and Applications 22 (1) (2013) 257-267.
[4] S Balasundaram, M Tanveer, Smooth Newton method for implicit Lagrangian twin support vector regression, International Journal of Knowledge-Based and Intelligent Engineering Systems 17 (4) (2013) 267-278.
[5] GEP Box, GM Jenkins, Time series analysis: Forecasting and Control, Holden-Day, San Francisco, 1976.
[6] PS Bradley, OL Mangasarian, Feature selection via concave minimization and support vector machines, In: Shavlik (Ed.), Machine learning proceedings of the Fifteenth International Conference, (1998) 82-90, San Francisco, MK.
[7] CC Chang, CJ Lin, LIBSVM: A library for support vector machines, (2011) http://www.csie.ntu.edu.tw/ cjlin/libsvm.
[8] S Chen, M Wang, Seeking multi-threshold directly from support vectors for image segmentation, Neurocomputing, 67 (2005) 335-344.
[9] X Chen, J Yang, J Liang, A flexible support vector machine for regression, Neural Computing & Applications 21 (8) (2012) 2005-2013.
[10] X Chen, J Yang, J Liang, Q Ye, Smooth twin support vector regression, Neural Computing & Applications 21 (3) (2012) 505-513.
[11] C Cortes, VN Vapnik, Support vector networks, Machine Learning 20 (1995) 273-297.
[12] N Cristianini, J Shawe-Taylor, An introduction to support vector machines and other kernel based learning method, Cambridge University Press, Cambridge, 2000.
[13] DELVE, Data for Evaluating Learning in Valid Experiments (2005). http://www.cs.toronto.edu/d̃elve/data
[14] T Ebrahimi, GN Garcia, JM Vesin, Joint time-frequency-space classification of EEG in a brain-computer interface application, Journal of Applied Signal Processing 1 (7) (2003) 713-729.
[15] AV Fiacco, GP McCormick, Nonlinear Programming: Sequential unconstrained minimization techniques, John Wiley & Sons, New York, 1968.
[16] G Fung, OL Mangasarian, Finite Newton method for Lagrangian support vector machine classification, Neurocomputing 55 (1-2) (2003) 39-55.
[17] G Fung, OL Mangasarian, A feature selection Newton method for support vector machine classification, Computational Optimization and Applications 28 (2) (2004) 185-202.
[18] S Gao, Q Ye, N Ye, 1-Norm least squares twin support vector machines. Neurocomputing 74 (2011): 3590-3597.
[19] GH Golub, CF Van Loan, Matrix computations, (3rd edition), The Johns Hopkins University Press, 1996.
[20] I Guyon, J Weston, S Barnhill, VN Vapnik, Gene selection for cancer classification using support vector machine, Machine Learning 46 (2002) 389-422.
[21] JB Hiriart-Urruty, JJ Strodiot, VH Nguyen, Generalized Hessian matrix and second order optimality conditions for problems with CL1 data, Applied Mathematics and Optimization 11 (1984) 43-56.
[22] Jayadeva, R Khemchandani, S Chandra, Twin support vector machines for pattern classification, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (5) (2007) 905-910.
[23] T Joachims, C Ndellec, Rouveriol, Text categorization with support vector machines: learning with many relevant features. In: European Conference on Machine Learning, Chemnitz, Germany 10 (1998) 137-142.
[24] MA Kumar, M Gopal, Least squares twin support vector machines for pattern classification, Expert Systems with Applications 36 (2009) 7535-7543.
[25] YJ Lee, OL Mangasarian, SSVM: A Smooth support vector machine for classification, Computational Optimization and Applications 20 (1) (2001) 5-22.
[26] OL Mangasarian, Exact 1-norm support vector machines via unconstrained convex differential minimization, Journal of Machine Learning Research 7 (2006) 1517-1530.
[27] OL Mangasarian, DR Musicant, Lagrangian support vector machines, Journal of Machine Learning Research 1 (2001) 161-177.
[28] OL Mangasarian, EW Wild, Multisurface proximal support vector classification via generalized eigenvalues, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (1) (2006) 69-74.
[29] S Mukherjee, E Osuna, F Girosi, Nonlinear prediction of chaotic time series using support vector machines, In: NNSP'97: Neural Networks for Signal Processing VII: in Proc. of IEEE Signal Processing Society Workshop, Amelia Island, FL, USA, (1997) 511-520.
[30] KR Muller, AJ Smola, G Ratsch, B Schlkopf, J Kohlmorgen, Using support vector machines for time series prediction, In: Schlkopf B, Burges CJC, Smola AJ (Eds.), Advances in Kernel Methods- Support Vector Learning MIT Press, Cambridge, MA, (1999) 243-254.

[31] PM Murphy, DW Aha, UCI repository of machine learning databases, (1992), University of California, Irvine. http://www.ics.uci.edu/ mlearn
[32] E Osuna, R Freund, F Girosi, Training support vector machines: An application to face detection. In: Proceedings of Computer Vision and Pattern Recognition, (1997) 130-136.
[33] X Peng, TSVR: An efficient twin support vector machine for regression, Neural Networks 23 (3) (2010) 365-372.
[34] X Peng, Building sparse twin support vector machine classifiers in primal space, Information Sciences 181 (2011) 3967-3980.
[35] WH Press, SA Teukolsky, WT Vetterling, BP Flannery, Numerical recipes in C, (2nd edition), Cambridge University Press, 1994.
[36] B Ribeiro, Kernelized based functions with Minkovsky's norm for SVM regression. In: Proceedings of the International Joint Conference on Neural Networks, IEEE press, (2002) 2198-2203.
[37] YH Shao, NY Deng, ZM Yang, WJ Chen, Z Wang, Probabilistic outputs for twin support vector machines, Knowledge-Based Systems 33 (2012) 145-151.
[38] YH Shao, CH Zhang, XB Wang, NY Deng, Improvements on twin support vector machines, IEEE Transactions on Neural Networks, 22 (6) (2011), 962-968.
[39] YH Shao, CH Zhang, ZM Yang, L Jing, NY Deng, An ϵ-twin support vector machine for regression, Neural Computing and Applications 23 (2012) 175-185.
[40] M Tanveer, Smoothing technique on linear programming twin support vector machines, International Journal of Machine Learning and Computing 3 (2) (2013) 240-244.
[41] M Tanveer, Robust and sparse linear programming twin support vector machines, Cognitive Computation, 7 (1) (2015) 137-149.
[42] M Tanveer, Application of smoothing techniques for linear programming twin support vector machines, Knowledge and Information Systems, 45 (1) (2015) 191-214.
[43] Y Tian, X Ju, Z Qi, Efficient sparse nonparallel support vector machines for classification, Neural Computing and Applications 24 (5) (2013) 1089-1099.
[44] AN Tikhonov, VY Arsen, Solutions of ill-posed problems, John Wiley & Sons, New York, 1977.
[45] VN Vapnik, Statistical Learning Theory, Wiley, New York, 1998.
[46] VN Vapnik, The nature of statistical learning theory, (2nd Edition), Springer, New York, 2000.
[47] Y Xu, L Wang, A weighted twin support vector regression, Knowledge-Based Systems (33) (2012) 92-101.
[48] Y Xu, L Wang, P Zhong, A rough margin-based ν-twin support vector machine, Neural Computing and Applications 21 (6) (2012) 1307-1317.
[49] P Zhong, Y Xu, Y Zhao, Training twin support vector regression via linear programming, Neural Computing and Applications 21 (2) (2012) 399-407.