On implicit Lagrangian twin support vector regression by Newton method

S. Balasundaram[*]
School of Computer and Systems Sciences
Jawaharlal Nehru University
New Delhi, 110067, India
E-mail:balajnu@gmail.com

Deepak Gupta
School of Computer and Systems Sciences
Jawaharlal Nehru University
New Delhi, 110067, India
E-mail:deepakjnu85@gmail.com

Received 5 February 2012

Accepted 9 June 2013

Abstract

In this work, an implicit Lagrangian for the dual twin support vector regression is proposed. Our formulation leads to determining non-parallel ε–insensitive down- and up- bound functions for the unknown regressor by constructing two unconstrained quadratic programming problems of smaller size, instead of a single large one as in the standard support vector regression (SVR). The two related support vector machine type problems are solved 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 SVR. Similar or better generalization performance of the proposed method clearly illustrates its effectiveness and applicability.

Keywords: Implicit Lagrangian support vector machines, Non parallel planes, Support vector regression, Twin support vector regression

1. Introduction

Support vector machine (SVM) which is based on structural risk minimization (SRM) induction principle is an excellent kernel-based machine learning tool. It has been successfully applied to classification and regression problems of practical importance such as face recognition[1], biomedicine[2], image segmentation[3] and time series forecasting[4,5] . It is a supervised learning method where the training leads to solving a quadratic programming problem (QPP) having unique optimal solution. Further, since it results in better generalization performance over other learning methods such as artificial neural networks (ANNs), it becomes one of the most attractive methods used in machine learning.

The central idea of SVM for classification is to construct a surface that partitions the input samples from different classes with maximum margin[6,7]. Recently, a new method called multi-surface proximal SVM was proposed[8] wherein two non-parallel planes are constructed so that each one of them is closest to points of its own class meanwhile as far away as possible from points of other class. This turns out to solving two generalized eigenvalue problems whose solutions become the eigenvectors corresponding to their smallest eigenvalues. Similar in sprit to this idea, twin support vector machine (TWSVM) was proposed in Ref.9 wherein two non-parallel planes are constructed by solving a pair of QPPs of smaller size in comparison to the standard SVM. TWSVM has attracted much attention due to its low computational cost and better generalization classification ability over the standard SVM[9-11].

The goal of regression problem is in determining the functional relationship between the inputs and their

[*]Corresponding author: balajnu@gmail.com

corresponding observed values. Originally, SVMs have been developed for classification problems and with the introduction of ε-insensitive error loss function by Vapnik[7] it has been extended to regression problems. Support vector regression (SVR) formulation leads to solving a QPP subject to linear inequality constraints[6,7] and it exhibits excellent performance in many fields of importance such as time series prediction[4,5] and optimal control[12]. For the study on "equivalent" SVR formulations in 2-norm instead of the usual 1-norm, the interested reader is referred to Refs. 13-16.

Recently, Peng[17] proposed twin SVR (TSVR) similar in sprit to TWSVM[9] wherein a pair of nonparallel functions corresponding to the ε-insensitive down- and up-bounds are determined. This formulation leads to solving two related SVM-type of problems, i.e. 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[17]. 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 in Ref.18 as an unconstrained minimization problem and further solved 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 Ref.19.

Motivated by the studies on implicit Lagrangian SVM for classification[20] and regression[21], in this work, we propose their extension on TSVR. Our formulation leads to solving a pair of unconstrained QPPs of smaller size in dual variables and their solutions are obtained by Newton method. Numerical experiments were performed on a number of interesting synthetic and real-world datasets and their results were compared with SVR and TSVR to verify the effectiveness of the proposed method.

In this work, all vectors are assumed 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. When x is orthogonal to y, we write $x \perp y$. For $x \in R^n$, the plus function x_+ is defined as: $(x_+)_i = \max\{0, x_i\}$, where $i = 1,\ldots,n$. The 2-norm of a vector x and a matrix Q will be denoted by $\| x \|$ and $\| Q \|$ respectively. We denote the vector of ones of dimension m by e and the identity matrix of appropriate size by I.

The paper is organized as follows. Section 2 dwells briefly SVR and TSVR. In Section 3 we begin with TSVR formulation and its dual in 2-norm and then formulate the implicit Lagrangian TSVR (LTSVR) as a pair of unconstrained minimization problems whose solutions are obtained using Newton method. Numerical experiments were performed and their results were compared with that of SVR and TSVR in Section 4 and finally we conclude our work in Section 5.

2. Background

In this section, we briefly describe the standard SVR and twin SVR formulations.

Assume that a training set $\{(x_i, y_i)\}_{i=1,2,\ldots,m}$ of m points in n-dimensional input space R^n be given such that for each input example $x_i \in R^n$, let $y_i \in R$ be its corresponding observed value. Throughout in this work we assume that the input examples are represented 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: $y = (y_1,\ldots, y_m)^t$.

2.1. Support vector regression formulation

The goal of the standard ε–insensitive error loss SVR problem is in determining the regression estimation function $f : R^n \longrightarrow R$ by mapping the input examples into a higher dimensional feature space via a feature map $\varphi(.)$

and learning a linear regression function in the feature space, i.e. we will have:

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

where w is a vector in the feature space and b is a scalar. This problem can be formulated as a constrained minimization problem defined by[6,7] :

$$\min_{w,b,\xi_1,\xi_2} \frac{1}{2} w^t w + C(e^t \xi_1 + e^t \xi_2)$$

subject to

$$y - (\varphi(A)w + eb) \le \varepsilon e + \xi_1,$$

$$(\varphi(A)w + eb) - y \le \varepsilon e + \xi_2$$

and $\quad \xi_1, \xi_2 \ge 0, \qquad (2)$

where ξ_1, ξ_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: $u_1 = (u_{11},...,u_{1m})^t, u_2 = (u_{21},...,u_{2m})^t \in R^m$, one can show that the Wolfe dual of (2) can be obtained in the following form:

$$\min_{u_1,u_2} \frac{1}{2} \sum_{i,j=1}^{m} (u_{1i} - u_{2i}) \varphi(x_i)^t \varphi(x_j)(u_{1j} - u_{2j})$$
$$+ \varepsilon \sum_{i=1}^{m} (u_{1i} + u_{2i}) - \sum_{i=1}^{m} y_i (u_{1i} - u_{2i})$$

subject to

$$\sum_{i=1}^{m} (u_{1i} - u_{2i}) = 0 \text{ and } 0 \le u_1, u_2 \le Ce .$$

Now applying the kernel trick[6,7], the dual problem can be rewritten as:

$$\min_{u_1,u_2} \frac{1}{2} \sum_{i,j=1}^{m} (u_{1i} - u_{2i}) k(x_i, x_j)(u_{1j} - u_{2j})$$
$$+ \varepsilon \sum_{i=1}^{m} (u_{1i} + u_{2i}) - \sum_{i=1}^{m} y_i (u_{1i} - u_{2i})$$

subject to

$$\sum_{i=1}^{m} (u_{1i} - u_{2i}) = 0 \text{ and } 0 \le u_1, u_2 \le Ce ,$$

where $k(.,.)$ is a kernel function. In this work, we considered the Gaussian kernel function defined by:

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

and $\sigma > 0$ is a parameter.

Finally, for any input example $x \in R^n$ its prediction by the decision function $f(.)$ is given by[6,7] :

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

2.2. Twin support vector regression (TSVR)

Unlike in SVR where a single regression estimation function is learned to fit the given inputs, TSVR[17] finds two non-parallel functions corresponding to the ε–insensitive down- and up-bounds. This formulation has the advantage of solving a pair of QPPs each having m number of constraints instead of a single QPP with $2m$ number of constraints as in SVR, where m is the number of input examples.

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

$$f_1(x) = w_1^t x + b_1 \text{ and } f_2(x) = w_2^t x + b_2 , \tag{3}$$

respectively, are determined 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[17] :

$$\min_{(w_1,b_1,\xi_1)\in R^{n+1+m}} \frac{1}{2}\| y-\varepsilon_1 e-(Aw_1+b_1 e)\|^2 + C_1 e^t \xi_1$$

subject to

$$y-(Aw_1+b_1 e) \geq \varepsilon_1 e-\xi_1 \ , \ \xi_1 \geq 0 \qquad (4)$$

and

$$\min_{(w_2,b_2,\xi_2)\in R^{n+1+m}} \frac{1}{2}\| y+\varepsilon_2 e-(Aw_2+b_2 e)\|^2 + C_2 e^t \xi_2$$

subject to

$$(Aw_2+b_2 e)-y \geq \varepsilon_2 e-\xi_2 \ , \quad \xi_2 \geq 0 \, . \qquad (5)$$

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

By introducing Lagrange multipliers: $u_1=(u_{11},...,u_{1m})^t$, $u_2=(u_{21},...,u_{2m})^t \in R^m$, one can show that the Wolfe duals of (4) and (5) can be obtained as a pair of QPPs of the following form[17] :

$$\min_{u_1\in R^m} \frac{1}{2}u_1^t \ G(G^tG)^{-1}G^t \ u_1 - (y-\varepsilon_1 e)^t(\ G(G^tG)^{-1}G^t - I \ )u_1$$

subject to

$$0 \leq u_1 \leq C_1 e \qquad (6)$$

and

$$\min_{u_2\in R^m} \frac{1}{2}u_2^t \ G(G^tG)^{-1}G^t \ u_2 - (y+\varepsilon_2 e)^t(I - G(G^tG)^{-1}G^t \ )u_2$$

subject to

$$0 \leq u_2 \leq C_2 e, \qquad (7)$$

respectively. The down- and up- bound estimation functions (3) can be obtained from the solutions of the dual problems (6) and (7) using the following relations:

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

and

$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = (G^tG)^{-1}G^t \ (y+\varepsilon_2 e+u_2), \qquad (8)$$

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

Finally, the end regressor $f: R^n \to R$ is obtained using $f_1(.)$ and $f_2(.)$ as:

$$f(x)=\frac{1}{2}(f_1(x)+f_2(x)) \text{ for all } x\in R^n \qquad (9)$$

To generalize our results to nonlinear regressors, we employ the kernel technique followed in Ref.22.

For the input matrix $A\in R^{m\times n}$ and a nonlinear kernel function $k(.,.)$ given, consider the kernel matrix $K=K(A,A^t)$ of order *m* whose (i,j)-th element is defined by: $K(A,A^t)_{ij}=k(x_i,x_j)\in R.$ Further, let $K(x^t,A^t)= \ (k(x,x_1),...,k(x,x_m))$ be a row vector in R^m.

Consider the kernel generated down- and up-bounds $f_1(.)$ and $f_2(.)$, defined by: for any $x\in R^n$,

$$f_1(x)=K(x^t,A^t)w_1+b_1$$

and

$$f_2(x)=K(x^t,A^t)w_2+b_2 \, . \qquad (10)$$

They will be determined using the solutions of the following two constrained minimization problems given by[17]:

$$\min_{(w_1,b_1,\xi_1)\in R^{m+1+m}} \frac{1}{2}\| y-\varepsilon_1 e-(K(A,A^t)w_1+b_1 e)\|^2 + C_1 e^t \xi_1$$

subject to

$$y-(K(A,A^t)w_1+b_1e)\geq \varepsilon_1 e-\xi_1 ,$$
$$\xi_1 \geq 0 \tag{11}$$

and

$$\min_{(w_2,b_2,\xi_2)\in R^{m+1+m}} \frac{1}{2}\| y+\varepsilon_2 e-(K(A,A^t)w_2+b_2e)\|^2 + C_2 e^t \xi_2$$

subject to

$$(K(A,A^t)w_2+b_2e)-y\geq \varepsilon_2 e-\xi_2 ,$$
$$\xi_2 \geq 0 , \tag{12}$$

where ξ_1, $\xi_2 \in R^m$ are slack variable vectors and $C_1, C_2 > 0$; $\varepsilon_1, \varepsilon_2 > 0$ are input parameters. Note that when the linear kernel function is applied, the kernel TSVR will reduce to the linear TSVR[17].

Proceeding as in the linear case, the pair of dual QPPs for the kernel TSVR can be obtained. In fact, the two QPPs for the kernel TSVR are found to be exactly the same as (6) and (7) satisfying (8) but the augmented matrix G will become: $G=[K(A,A^t) \quad e]$. Finally the end regressor defined by (9) is obtained using (8) and (10). For a detailed discussion on the method of solution and numerical experiments of TSVR, see Ref.17.

3. Implicit Lagrangian twin support vector regression (LTSVR)

In this section, the implicit Lagrangian formulation for the TSVR in its dual is proposed as an extension of the work of Ref.20 initially suggested for classification problems. Our formulation leads to solving two unconstrained minimization problems whose solutions will be obtained using Newton method.

Consider the primal TSVR in 2-norm as a pair of constrained minimization problems of the form:

$$\min_{(w_1,b_1,\xi_1)\in R^{n+1+m}} \frac{1}{2}\| y-\varepsilon_1 e-(Aw_1+b_1e)\|^2 + \frac{C_1}{2}\xi_1^t \xi_1$$

subject to

$$y-(Aw_1+b_1e)\geq \varepsilon_1 e-\xi_1 \tag{13}$$

and

$$\min_{(w_2,b_2,\xi_2)\in R^{n+1+m}} \frac{1}{2}\| y+\varepsilon_2 e-(Aw_2+b_2e)\|^2 + \frac{C_2}{2}\xi_2^t \xi_2$$

subject to

$$(Aw_2+b_2e)-y\geq \varepsilon_2 e-\xi_2 \tag{14}$$

where ξ_1, ξ_2 are vectors of slack variables and $C_1, C_2 > 0$; $\varepsilon_1, \varepsilon_2 > 0$ are input parameters. Since the non-negativity constraints of the slack variables ξ_1 and ξ_2 will be satisfied automatically at optimality, they are no longer necessary to be considered explicitly in (13) and (14). Using the solutions of (13) and (14), the down- and up- bound regressors $f_1(.)$ and $f_2(.)$ defined by (3) will be determined.

The Wolfe duals corresponding to the pair of primal problems (13) and (14) can be written as a pair of QPPs of the following form:

$$\min_{0\leq u_1\in R^m} \frac{1}{2}(y-\varepsilon_1 e)^t\, G(G^tG)^{-1}G^t\,(y-\varepsilon_1 e) -\frac{1}{2}\| y-\varepsilon_1 e\|^2 +\frac{1}{2}u_1^t(\frac{I}{C_1}+G(G^tG)^{-1}G^t\,)u_1 -(y-\varepsilon_1 e)^t(\,G(G^tG)^{-1}G^t-I\,)u_1 \tag{15}$$

and

$$\min_{0\leq u_2\in R^m} \frac{1}{2}(y+\varepsilon_2 e)^t\, G(G^tG)^{-1}G^t\,(y+\varepsilon_2 e) -\frac{1}{2}\| y+\varepsilon_2 e\|^2 +\frac{1}{2}u_2^t(\frac{I}{C_2}+G(G^tG)^{-1}G^t\,)u_2 -(y+\varepsilon_2 e)^t(\,I-G(G^tG)^{-1}G^t\,)u_2, \tag{16}$$

respectively, where $u_1,\ u_2 \in R^m$ are Lagrange multipliers and $G = [A \quad e]$ is an augmented matrix such that (8) holds.

Define the matrix:

$$H = G(G^t G)^{-1} G^t . \qquad (17)$$

Now, dropping the terms which are independent of the dual variables and substituting (17) in the dual formulations (15) and (16), TSVR in dual can be derived as a pair of minimization problems of the form:

$$\min_{0 \le u_1 \in R^m} L_1(u_1) = \frac{1}{2} u_1^t Q_1 u_1 - r_1^t u_1$$

and

$$\min_{0 \le u_2 \in R^m} L_2(u_2) = \frac{1}{2} u_2^t Q_2 u_2 - r_2^t u_2 , \qquad (18)$$

where

$$Q_1 = \frac{I}{C_1} + H , \qquad Q_2 = \frac{I}{C_2} + H ,$$

$$r_1 = (\ H - I\ )(y - \varepsilon_1 e)$$

and

$$r_2 = (\ I - H\ )(y + \varepsilon_2 e) . \qquad (19)$$

Finally, the end regressor $f(.)$ will be determined using (3), (8), (9) and the solutions of (18).

The nonlinear TSVR in primal can be constructed as a pair of QPPs in 2-norm as:

$$\min_{(w_1, b_1, \xi_1) \in R^{m+1+m}} \frac{1}{2} \| y - \varepsilon_1 e - (K(A, A^t) w_1 + b_1 e) \|^2 + \frac{C_1}{2} \xi_1^t \xi_1$$

subject to

$$y - (K(A, A^t) w_1 + b_1 e) \ge \varepsilon_1 e - \xi_1 \qquad (20)$$

and

$$\min_{(w_2, b_2, \xi_2) \in R^{m+1+m}} \frac{1}{2} \| y + \varepsilon_2 e - (K(A, A^t) w_2 + b_2 e) \|^2 + \frac{C_2}{2} \xi_2^t \xi_2$$

subject to

$$(K(A, A^t) w_2 + b_2 e) - y \ge \varepsilon_2 e - \xi_2 . \qquad (21)$$

Further, defining the augmented matrix G as: $G = [K(A, A^t) \ \ e]$, the pair of dual QPPs corresponding to (20) and (21) can be formulated as a pair of minimization problems of the form (18) where $u_1, u_2 \in R^m$ are Lagrange multipliers and Q_1, Q_2, r_1, r_2 are computed using (17) and (19). In this case, the $\varepsilon -$insensitive down- and up-kernel generated regressors $f_1(.)$ and $f_2(.)$ will become: for any vector $x \in R^n$,

$$f_1(x) = [K(x^t, A^t) \ \ 1]\ (G^t G)^{-1} G^t \ (y - \varepsilon_1 e - u_1)$$

and

$$f_2(x) = [K(x^t, A^t) \ \ 1]\ (G^t G)^{-1} G^t \ (y + \varepsilon_2 e + u_2),$$

respectively. Finally, taking the mean of $f_1(.)$ and $f_2(.)$ the kernel regressor $f : R^n \to R$ is obtained.

Applying the Karush-Kuhn-Tucker (KKT) necessary and sufficient optimal conditions for the dual TSVR will lead to the following pair of classical complementarity problems[23]:

$$0 \le u_1 \perp (Q_1 u_1 - r_1) \ge 0$$

and

$$0 \le u_2 \perp (Q_2 u_2 - r_2) \ge 0 . \qquad (22)$$

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

$$0 \le u \perp v \ge 0 \text{ if and only if } u = (u - \alpha v)_+$$

for any $\alpha > 0$, the solutions of the following equivalent pair of problems will be considered[22]: for any $\alpha_1, \alpha_2 > 0$,

$$(Q_1u_1 - r_1) = (Q_1u_1 - \alpha_1u_1 - r_1)_+$$

and

$$(Q_2u_2 - r_2) = (Q_2u_2 - \alpha_2u_2 - r_2)_+ . \quad (23)$$

It turns out that conditions (23) become necessary and sufficient to be satisfied by the unconstrained minimum of the following pair of implicit Lagrangians[24] associated to the pair of dual problems (18): for any $\alpha_1, \alpha_2 > 0$,

$$\min_{u\in R^m} L_1(u) = \frac{1}{2}u^tQ_1u - r_1^tu$$
$$+\frac{1}{2\alpha_1}(\| (Q_1u - \alpha_1u - r_1)_+ \|^2 - \| Q_1u - r_1 \|^2) \quad (24)$$

and

$$\min_{u\in R^m} L_2(u) = \frac{1}{2}u^tQ_2u - r_2^tu$$
$$+\frac{1}{2\alpha_2}(\| (Q_2u - \alpha_2u - r_2)_+ \|^2 - \| Q_2u - r_2 \|^2)$$

. (25)

In this work we solve the pair of dual QPPs defined by (24) and (25) using the well-known Newton method[20].

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

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

where *i=0, 1, 2, …*

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

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

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 Ref.25 exists and is given by: for all $u \in R^m$,

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

where *diag(.)* is a diagonal matrix of order *m.*

Using the property that the matrix $Q_k \in R^{m\times m}$ defined by (19) is positive-definite and choosing the parameter α_k satisfying the condition[20]:

$$\alpha_k > \| Q_k \| \text{ where } k = 1,2,$$

the Newton's iterative step can be rewritten in the following simpler form: for *i=0, 1,2 …*

$$u^{i+1} = u^i - \partial h_k(u^i)^{-1} h_k(u^i), \quad (26)$$

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

$$h_k(u) = (Q_ku - r_k) - (Q_ku - \alpha_ku \ - r_k)_+$$

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

$$\partial h_k(u) = (Q_k$$
$$+ diag((Q_ku - \alpha_ku - r_k)_*)(\alpha_k I - Q_k)).$$

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

$$D_k = diag((Q_ku - \alpha_ku - r_k)_*),$$

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

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

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

Now, we state the Newton iterative Algorithm with Armijo stepsize[14,15,20] for solving each of the unconstrained minimization problems defined by (24) and (25).

```
Newton Algorithm with Armijo stepsize
for solving (24),(25) with $k = 1,2$

Start with 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 equations
        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 the stepsize $\lambda_i = \max\{1, \frac{1}{2}, \frac{1}{4}, \ldots\}$ is
     such that:
        $L_k(u^i) - L_k(u^i + \lambda_i d^i) \geq -\delta \lambda_i h_k(u^i)^t d^i$
        and $\delta \in (0, \frac{1}{2})$.
```

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 Refs. 20, 21.

Note that the matrix $G^t G$ 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 βI the modified matrix $(\beta I + G^t G)$ will be used, where β is chosen to be a very small positive number.

4. Numerical experiments and comparison of results

In order to investigate the performance of the proposed implicit LTSVR, we performed experiments on a number of interesting synthetic and real-world datasets and compared the results with SVR and TSVR. In Table 1, the list of functions considered for generating synthetic datasets were summarized. Also we carried out experiments on several real-world datasets- the Box Jenkins gas furnace data; Google, IBM, Intel, RedHat, Microsoft and S&P 500 financial time series data; Sunspots 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 were coded in MATLAB R2009a running 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 using (26), i.e. without Armijo stepsize. TSVR was implemented using optimization tool box of MATLAB and for the implementation of SVR we used MOSEK optimization tools for MATLAB available at http://www.mosek.com. In all the experiments considered, Gaussian nonlinear kernel function was chosen. 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 σ from the sets $\{10^{-3}, \ldots, 10^{3}\}$ and $\{2^{-10}, 2^{-9}, \ldots, 2^{10}\}$ respectively and by choosing these optimal parameter values for training, the RMSE on the test set was calculated.

4.1 *Synthetic datasets*

To demonstrate the performance of the proposed implicit LTSVR on synthetic datasets, first we consider the following function for approximation[26]:

$$y(x) = \frac{4}{|x|+2} + \cos(2x) + \sin(3x),$$

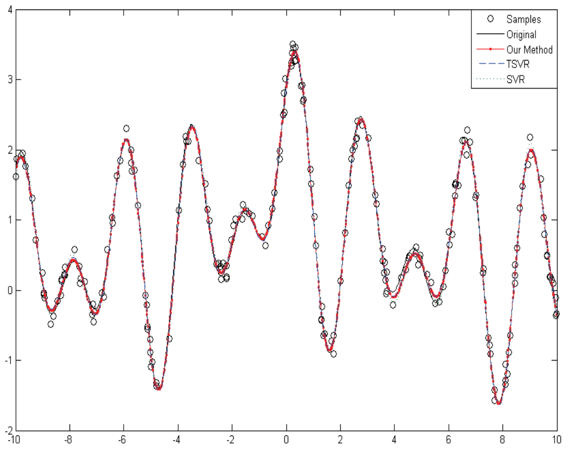
for $x \in [-10,10]$.

We selected, randomly, 200 points for training and 1000 points for testing under uniform distribution over the interval $[-10,10]$. 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 SVR, TSVR and the implicit LTSVR for uniform and Gaussian additive noises were illustrated in Figure 1a and Figure 1b respectively along with the exact function. The noisy training samples are indicated by the symbol 'o'. The results obtained on the test set were summarized in Table 2.

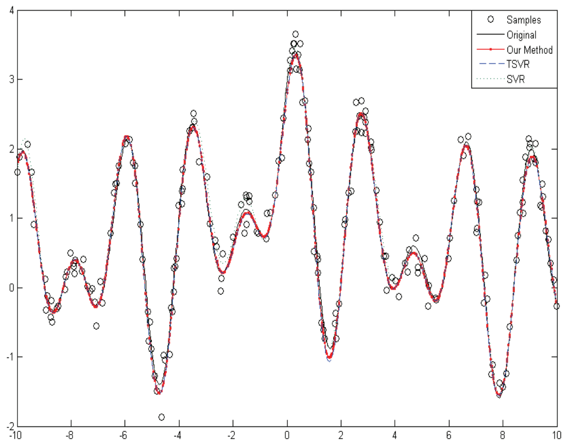
To further analyze the performance of the proposed method, another 5 synthetic datasets of 1200 samples each were generated using functions whose definitions are given in Table 1. As in the previous case, 200 samples were used for training and 1000 samples for testing. The observed value y_i for the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

$$y_i = f(x_i) + \varsigma_i, \quad \text{for } i = 1,\dots,m,$$

where the additive noise ς_i was sampled in two ways: (i). uniform distribution over the interval $[-0.2,0.2]$; (ii). Gaussian distribution with mean = 0 and standard deviation = 0.2. 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.



a).Uniform noise over the interval [-0.2, 0.2].



b). Gaussian noise with mean zero and standard deviation 0.2.

Figure 1: Results of approximation of $\frac{4}{|x|+2} + \cos(2x) + \sin(3x)$ with our method, SVR and TSVR when different kinds of additive noise were used. Gaussian kernel was employed.

Table 1. Functions used for generating synthetic datasets.

Name	Function Definition	Domain of Definition
Function 1	$\frac{4}{\lvert x\rvert+2}+\cos(2x)+\sin(3x)$	$x\in[-10,10]$
Function 2	$\frac{4}{\lvert x\rvert+2}+\cos(2x)$	$x\in[-10,10]$
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 2. Performance comparison of the proposed method with SVR and TSVR on synthetic datasets for uniformly distributed and Gaussian noises. RMSE and training time (seconds) were used for comparison. Gaussian kernel was employed. The best result is shown in boldface.

	Uniformly distributed noise			Gaussian noise		
Datasets (Train size, Test size)	SVR RMSE (C, σ) Training time	TSVR RMSE $(C_1=C_2,\sigma)$ Training time	Our Method RMSE $(C_1=C_2,\sigma)$ Training time	SVR RMSE (C, σ) Training time	TSVR RMSE $(C_1=C_2,\sigma)$ Training time	Our Method RMSE $(C_1=C_2,\sigma)$ Training time
Function1 (200×1,1000×1)	**0.0397** $(10^3,2^{-1})$ 0.9449	0.0489 $(10^0,2^0)$ 0.2898	0.0463 $(10^1,2^0)$ 0.2338	**0.0585** $(10^{-1},2^1)$ 1.1812	0.0769 $(10^2,2^0)$ 0.2569	0.0644 $(10^1,2^0)$ 0.2355
Function2 (200×1,1000×1)	0.0485 $(10^3,2^2)$ 1.1613	**0.0463** $(10^1,2^0)$ 0.3011	0.0475 $(10^1,2^0)$ 0.2293	0.1110 $(10^3,2^0)$ 1.1414	0.0835 $(10^{-1},2^0)$ 0.1706	**0.0834** $(10^{-3},2^0)$ 0.2031
Function3 (200×5,1000×5)	0.0626 $(10^3,2^1)$ 1.1744	**0.0597** $(10^2,2^1)$ 0.3790	0.0609 $(10^1,2^1)$ 0.2295	0.1039 $(10^2,2^0)$ 1.1415	0.0782 $(10^1,2^1)$ 0.3238	**0.0700** $(10^1,2^1)$ 0.2259
Function4 (200×2,1000×2)	0.0822 $(10^1,2^{-1})$ 1.0515	**0.0121** $(10^1,2^0)$ 0.3724	0.014 $(10^1,2^0)$ 0.2265	0.0638 $(10^3,2^1)$ 0.9596	0.0338 $(10^{-1},2^0)$ 0.1487	**0.0327** $(10^0,2^0)$ 0.2378

Function5	0.0480	**0.0206**	0.0244	0.053	**0.0472**	0.0492
(200×2,1000×2)	$(10^3,2^1)$	$(10^1,2^{-3})$	$(10^1,2^{-3})$	$(10^2,2^0)$	$(10^0,2^{-2})$	$(10^0,2^{-2})$
	0.9807	0.3621	0.2272	0.9507	0.2899	0.2459
Function6	0.0338	**0.0228**	0.0553	0.0418	**0.0417**	0.0423
(200×2,1000×2)	$(10^1,2^0)$	$(10^1,2^{-1})$	$(10^1,2^0)$	$(10^3,2^{-2})$	$(10^1,2^0)$	$(10^1,2^0)$
	0.9440	0.4060	0.2335	0.9204	0.3136	0.2301

4.2. *Real-world datasets*

For all the real-world examples considered in this work, the original data is normalized in the following manner:

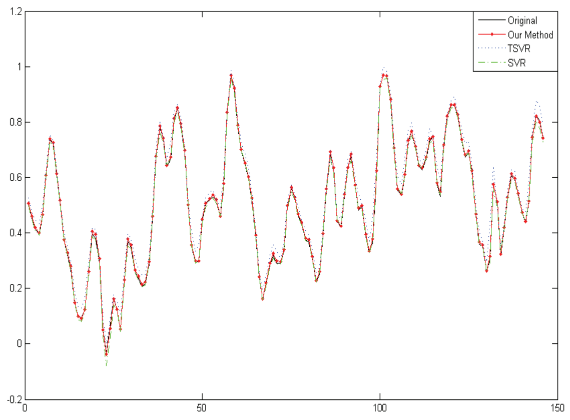
$$\bar{x}_{ij} = \frac{x_{ij} - x_j^{\min}}{x_j^{\max} - x_j^{\min}}$$

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.

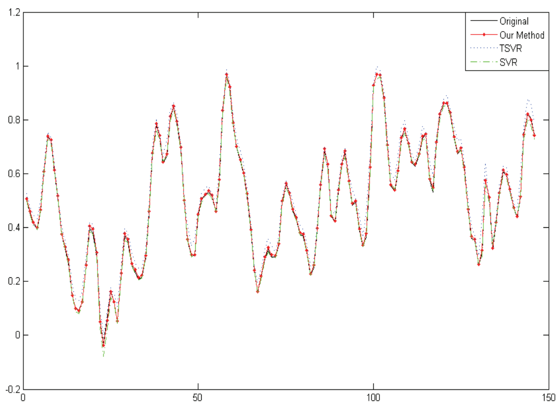
As the first real-world example, we considered the Box and Jenkins gas furnace data[27]. It consists of 296 input-output pairs of points of the form: $(u(t), y(t))$ where $u(t)$ is input gas flow rate whose output $y(t)$ is the CO_2 concentration from the gas furnace. The output $y(t)$ is predicted based on 6 attributes taken to be of the form[28]: $x(t) = (y(t-1), y(t-2), y(t-3), u(t-1), u(t-2), u(t-3))$. Thus, in total, we get 293 samples of the form: $(x(t), y(t))$. The odd samples were selected for training whereas the even samples were taken for testing. The performance of the proposed method on the training and test sets were shown in Figure 2a and Figure 2b respectively.

As examples of financial time series, we considered the datasets of Google, IBM, Intel, RedHat, Microsoft and S&P 500. They were taken from the website: http://dailyfinance.com. In all the examples considered, 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 rest for testing.

In addition, we compared implicit LTSVR with SVR and TSVR on few more well-known time series datasets. As in the financial time series datasets, five previous values



a). Prediction over the training set



b). Prediction over the test set

Figure 2: Results of comparison for gas furnace data of Box-Jenkins. Gaussian kernel was used.

were used to predict its current value.

The Sunspots 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 selected 100 samples for training and the rest for testing. The SantaFe-A time series dataset is available at: http://www-psych.stanford.edu/~andreas/Time-Series/SantaFe.html. It consists of 1000 values from which we get 995 samples in total. The first 300 samples were chosen for training and the rest for testing.

We performed our experiments on two more time series datasets generated by the Mackey-Glass time delay differential equation[4] given by

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

where the parameters *a, b* and time delay τ 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 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[29] given by

$$\dot{x} = \rho(y-x), \dot{y} = rx - y - xz$$

and $$\dot{z} = xy - bz ,$$

where ρ, r and b are input parameters. By taking different sampling rates, i.e. $\tau = 0.05$ & $\tau = 0.2$ and the parameter values: $\rho = 10$, $r = 28$, $b = 8/3$, two time series datasets $Lorenz_{0.05}$ and $Lorenz_{0.2}$ were generated. Each of them consists of 30000 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 in total. Among them the first 400 samples were chosen for training and the remaining samples for testing.

Finally, five more commonly used benchmark datasets- Machine CPU, Bodyfat, Concrete CS, Abalone and Kin-fh -were considered for testing the performance of the proposed implicit LTSVR.

The Machine CPU dataset consisting of 209 samples having 7 continuous features is taken from UCI repository[30]. 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 14 attributes. Here the observed value is the estimation of the body fat obtained from the body density values. The first 150 samples were taken for training and the rest for testing.

Table 3. Performance comparison of the proposed method with SVR and TSVR on real world datasets. RMSE and training time (seconds) were used for comparison. Gaussian kernel was employed. Bold type shows the best result.

Datasets (Train size, Test size)	SVR RMSE (C, σ) Training time	TSVR RMSE (C_1=C_2, σ) Training time	Our Method RMSE (C_1=C_2, σ) Training time
Gas furnace (147×6,146×6)	0.0410 (10^3,2^3) 0.5122	0.0355 (10^2,2^0) 0.1854	**0.0318** (10^1,2^0) 0.1243
Google (150×5,600×5)	**0.0270** (10^3,2^4) 0.5185	0.0285 (10^{-1},2^2) 0.1355	0.0299 (10^{-1},2^2) 0.1263

IBM (150×5,600×5)	0.0339 $(10^{1},2^{1})$ 0.517	0.0321 $(10^{-1},2^{1})$ 0.1415	**0.0317** $(10^{-3},2^{1})$ 0.1155
Intel (150×5,600×5)	**0.0320** $(10^{3},2^{4})$ 0.5242	0.0372 $(10^{0},2^{2})$ 0.2214	0.0429 $(10^{1},2^{2})$ 0.1308
RedHat (150×5,600×5)	**0.0343** $(10^{1},2^{1})$ 0.5256	0.0381 $(10^{0},2^{2})$ 0.1945	0.0358 $(10^{0},2^{2})$ 0.1262
Microsoft (150×5,600×5)	0.0330 $(10^{1},2^{1})$ 0.5243	0.0330 $(10^{-1},2^{1})$ 0.1409	**0.0328** $(10^{-3},2^{1})$ 0.1146
S&P500 (150×5,600×5)	**0.0276** $(10^{3},2^{2})$ 0.5225	0.0350 $(10^{-1},2^{1})$ 0.1513	0.0361 $(10^{-1},2^{1})$ 0.1255
Sunspots (100×5,190×5)	**0.0847** $(10^{2},2^{1})$ 0.2319	0.0859 $(10^{-1},2^{1})$ 0.084	0.0867 $(10^{0},2^{1})$ 0.0625
SantafeA (300×5,695×5)	**0.0417** $(10^{2},2^{-1})$ 2.3343	0.0473 $(10^{-1},2^{-2})$ 0.4787	0.0477 $(10^{1},2^{-2})$ 0.5599
Mg_{17} (400×5,1095×5)	0.0103 $(10^{2},2^{-1})$ 4.2819	**0.0069** $(10^{1},2^{-2})$ 1.1487	**0.0069** $(10^{1},2^{-2})$ 1.1165
Mg_{30} (400×5,1095×5)	0.0242 $(10^{2},2^{-1})$ 4.2059	**0.0226** $(10^{-2},2^{-2})$ 0.6254	0.0229 $(10^{-3},2^{-2})$ 0.9719
$Lorenz_{0.05}$ (400×5,2595×5)	0.0053 $(10^{3},2^{3})$ 7.0675	**0.0052** $(10^{1},2^{0})$ 2.4154	**0.0052** $(10^{1},2^{0})$ 1.7372
$Lorenz_{0.2}$ (400×5,2595×5)	0.0076 $(10^{3},2^{3})$ 7.2485	**0.0036** $(10^{-1},2^{-1})$ 1.5254	**0.0036** $(10^{1},2^{-1})$ 1.7529
Machine CPU (100×7,109×7)	0.0364 $(10^{3},2^{2})$ 0.2299	0.0434 $(10^{-1},2^{1})$ 0.0909	**0.0433** $(10^{1},2^{1})$ 0.0586
Bodyfat (150×14,102×14)	**0.0134** $(10^{3},2^{4})$ 0.5297	0.0221 $(10^{-1},2^{3})$ 0.1368	0.0456 $(10^{-1},2^{3})$ 0.1163
Concrete CS (700×8,330×8)	0.1607 $(10^{1},2^{0})$ 14.8498	**0.1606** $(10^{-2},2^{1})$ 1.8087	**0.1606** $(10^{-3},2^{1})$ 2.7539
Abalone (1000×8,3177×8)	0.1332 $(10^{3},2^{4})$ 33.8183	0.1189 $(10^{-1},2^{1})$ 4.8581	**0.1174** $(10^{0},2^{1})$ 7.973
Kin-fh (1000×32,7192×32)	0.0968 $(10^{3},2^{5})$ 34.2304	**0.0952** $(10^{-3},2^{3})$ 5.5199	**0.0952** $(10^{-3},2^{3})$ 6.1485

The Concrete CS dataset[30] contains 1030 samples having 8 features. The first 700 samples were used for training and the rest for testing. The goal of Abalone dataset[30] is the prediction of the ages of the abalones using 8 physical measurements as their features. We have chosen the first 1000 samples for training and the remaining 3177 samples for testing. The Kin-fh dataset[31] 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 selected for training and the other 7192 samples for testing.

The tenfold numerical results of the real-world datasets by SVR, TSVR and the implicit LTSVR along with the number of training and test samples chosen, the number of attributes, the parameter values and the training time were summarized in Table 3.

One can observe from Table 2 that the best accuracy for the proposed method is achieved only on 3 occasions among the 12 synthetic datasets considered. Still, it is very much comparable to the best results obtained in the remaining cases. Note that, increasing the error tolerance accuracy used to terminate Newton iterative algorithm may lead to further improved generalization accuracy. As of the training time, the proposed method spends the least CPU time in most cases.

Among the total of 18 real-world datasets, the proposed method achieves the best accuracy for 10 datasets in comparison to SVR and TSVR including 3 datasets whose training set size is above 700. This clearly illustrates that the generalization ability of the proposed method is as competitive as of the remaining methods. Regarding the training time, the proposed method shows impressive advantage on small datasets. It is obvious from Table 3 that the proposed method is always faster than SVR. However, when the size of the training set is above 400, it lost its superiority to TSVR.

It should be noted that both SVR and TSVR were solved by optimization packages which implement fast algorithms whereas the proposed method solves matrix equations inside while loops.

5. Conclusions

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 needed. 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.

Acknowledgments

The authors are extremely thankful to the referees whose valuable comments have greatly improved the earlier version of the paper.

References

1. 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.
2. 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.
3. Chen, S., & Wang, M. (2005). Seeking multi-threshold directly from support vectors for image segmentation, *Neurocomputing*, 67, 335-344.
4. 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.
5. 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.
6. Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines and kernel based learning methods,* Cambridge University Press.
7. Vapnik, V.N. (2000). *The nature of statistical learning theory*, 2nd Edition, Springer, New York.
8. Mangasarian, O.L., & Wild, E.W. (2006). Multisurface proximal support vector classification via generalized eigenvalues, *IEEE Transactions on*

Pattern Analysis and Machine Intelligence 28(1), pp.69-74.

9. 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.
10. Kumar M.A., & Gopal, M. (2008). Application of smoothing technique on twin support vector machines, *Pattern Recognition Letters*, 29, 1842-1848.
11. Kumar M.A., & Gopal, M. (2009). Least squares twin support vector machines for pattern classification, *Expert System with Applications*, 36, 7535-7543.
12. Suykens, J.A.K., Vandewalle, J., & Moor, B.D. (2001). Optimal control by least squares support vector machines, *Neural Networks,* 14(1), 23-25.
13. Balasundaram, S., & Kapil (2010). On Lagrangian support vector regression, *Expert Systems with Applications*, 37, 8784-8792.
14. Balasundaram, S., & Rampal Singh (2010). On finite Newton method for support vector regression, *Neural Computing & Applications,* 19, 967-977.
15. Lee, Y.J., Hsieh, W.-F., & Huang, C.-M. (2005). ε-SSVR: A smooth support vector machine for ε- insensitive regression, *IEEE Transactions on Knowledge and Data Engineering*, 17(5), 678-685.
16. Musicant, D.R., & Feinberg, A. (2004). Active set support vector regression, *IEEE Transactions on Neural Networks,* 15(2), 268-275.
17. Peng, X. (2010). TSVR: An efficient Twin Support Vector Machine for regression, *Neural Networks*, 23(3), 365-372.
18. Chen X., Yang J., Liang J., & Ye, Q. (2012). Smooth twin support vector regression, *Neural Computing & Applications*, 21, 505-513.
19. Zhong, P., Xu, Y., & Zhao, Y. (2012). Training twin support vector regression via linear programming, *Neural Computing & Applications*, 21, 399-407.
20. Fung, G., & Mangasarian, O.L. (2003). Finite Newton method for Lagrangian support vector machine, *Neurocomputing,* 55, 39-55.
21. Balasundaram, S., & Kapil (2011). Finite Newton method for implicit Lagrangian support vector regression, *International Journal of Knowledge based Intelligent Engineering Systems*, 15, 203-214.
22. Mangasarian, O.L., & Musicant, D.R. (2001). Lagrangian support vector machines, *Journal of Machine Learning Research,* 1, 161-177.
23. Mangasarian, O.L. (1994). *Nonlinear programming,* SIAM Philadelphia, PA.
24. Mangasarian, O.L., & Solodov, M.V. (1993). Nonlinear complementarity as unconstrained and constrained minimization, *Mathematical Programming, Series B*, 62, 277-297.
25. Hiriart-Urruty, J.-B., Strodiot, 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.
26. Li, G., Wen, C., Huang, G.-B., & Chen, Y. (2011). Error tolerance based support vector machine for regression, *Neurocomputing*, 74(5), 771-782.
27. Box, G.E.P., & Jenkins, G.M. (1976). *Time series analysis: Forecasting and Control*, Holden-Day, San Francisco.
28. 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.
29. Casdagli, M. (1989). Nonlinear prediction of chaotic time series, *Physica D*, 35, 335-356.
30. Murphy, P.M., & Aha, D.W. (1992). UCI repository of machine learning databases. University of California, Irvine. http://www.ics.uci.edu/~mlearn
31. DELVE, (2005). Data for Evaluating Learning in Valid Experiments, http://www.cs.toronto.edu/~delve/data