Int. J. Mach. Learn. & Cyber. (2015) 6:1029–1040
DOI 10.1007/s13042-015-0414-x

ORIGINAL ARTICLE

Newton method for implicit Lagrangian twin support vector machines

M. Tanveer[1]

Received: 13 October 2014 / Accepted: 7 August 2015 / Published online: 28 August 2015

Abstract In this paper, we proposed an implicit Lagrangian twin support vector machine (TWSVM) classifiers by formulating a pair of unconstrained minimization problems (UMPs) in dual variables whose solutions will be obtained using finite Newton method. The advantage of considering the generalized Hessian approach for our modified UMPs reduces to solving just two systems of linear equations as opposed to solving two quadratic programming problems in TWSVM and TBSVM, which leads to extremely simple and fast algorithm. Unlike the classical TWSVM and least square TWSVM (LSTWSVM), the structural risk minimization principle is implemented by adding regularization term in the primal problems of our proposed algorithm. This embodies the essence of statistical learning theory. Computational comparisons of our proposed method against GEPSVM, TWSVM, STWSVM and LSTWSVM have been made on both synthetic and well-known real world benchmark datasets. Experimental results show that our method yields significantly better generalization performance in both computational time and classification accuracy.

Keywords Machine learning · Nonparallel planes · Lagrangian support vector machines · Twin support vector machines · Unconstrained convex minimization

1 Introduction

Support vector machine (SVM), introduced by Vapnik and coworkers [6, 44], is an excellent kernel based tool for binary data classification problems. Over the past decades, SVM has played an important role in solving problems emerged in pattern recognition and machine learning community due to its novel state of art technique. Its applications include a wide spectrum of research areas, ranging from pattern recognition [30], text categorization [16], biomedicine [5] etc. They have shown excellent performance on wide variety of problems [15, 30] due to its method of constructing a hyper plane such that the band between the two hyper planes separate both the classes and distance between two hyper planes is maximized, leading to the introduction of regularization term. Unlike other machine learning methods such as artificial neural networks (ANNs), training of SVMs leads to solving a linearly constrained quadratic programming problem (QPP) having unique optimal solution. Combined with the advantage of having unique optimal solution and better generalization performance, SVM becomes one of the most popular methods for solving classification problems. One of the main challenges for SVM is the large computational cost that is associated with the quadratic programming problems (QPPs). To reduce the learning complexity of SVM, various algorithms with comparable classification abilities have been reported, see [2, 6, 15, 19, 20, 24, 33, 43]. Specifically, Fung and Mangasarian [11] proposed an implicit Lagrangian formulation for SVM and solved using

This work was carried out at the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India, and Department of Computer Science and Engineering, The LNM Institute of Information Technology (LNMIIT) Jaipur, INDIA.

✉ M. Tanveer
tanveergouri@gmail.com; mtanveer@ntu.edu.sg

1 School of Computer Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798, Singapore

Newton method. The presented method is simple, fast and can be applied to problems with large scale data.

To improve the computational speed, Jayadeva et al. [14] proposed a twin support vector machine (TWSVM) [14] for the binary classification data in the spirit of the proximal SVM [10] and generalized eigenvalue proximal support vector machine (GEPSVM) [26]. TWSVM generates two nonparallel hyper planes by solving a pair of smaller-sized QPPs such that each plane is closest to one of the classes and as far as possible from the other class. A fundamental difference between TWSVM and SVM is that TWSVM solves two small QPPs rather than solving one large QPP makes the learning speed of TWSVM approximately four times faster than that of the standard SVM. Now, TWSVM has become popular and widely used because of its low computational complexity. Recently, some scholars proposed variants of TWSVM to reduce the time complexity and keep the effectiveness of TWSVM, see [3, 4, 12, 17, 29, 31, 32, 34, 37–39, 41, 42, 45, 46]. Specifically, least square twin SVM (LSTWSVM) [18] has been proposed by using the squared loss function instead of the hinge one in TWSVM, leading to very fast training speed since two QPPs are replaced by two systems of linear equations.

One of the principle advantages of SVM is the implementation of structural risk minimization (SRM) principle [36]. However, in the primal formulations of TWSVM and LSTWSVM [18], only the empirical risk is minimized. Also, we notice that the inverse of G^tG appears in the dual formulation of TWSVM. Using the extra regularization term G^tG is nonsingular. This is not perfect way from the theoretical point of view although it has been handled by modifying the dual problems technically and elegantly [37]. Recently, Shao et al. [37] proposed twin bounded support vector machines (TBSVM) based on TWSVM. Unlike TWSVM, the SRM principle is implemented in TBSVM and SOR technique is applied to speed-up the computational time.

Motivated by the works of [11, 24, 37], we proposed in this paper an implicit Lagrangian formulation for the twin support vector machine (TWSVM) classifiers by formulating a pair of unconstrained minimization problems in dual variables whose solutions will be obtained using finite Newton method. There are some differences in our formulation and TWSVM. The idea of our formulation is to reformulate TWSVM as a strongly convex problem by incorporated regularization techniques to improve the training speed and robustness. 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 and TBSVM, which leads to extremely simple and fast algorithm. It is also worth mentioning that the proposed formulation does not need any specialized optimization packages. The results of experiments conducted on both artificial and publicly available benchmark datasets confirm its efficacy and feasibility.

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^ty, where x^t is the transpose of x. Whenever x is orthogonal to y, we write $x \perp y$. For $x = (x_1, x_2, \ldots, x_n)^t \in R^n$, the plus function x_+ is defined as: $(x_+)_i = \max\{0, x_i\}$, where $i = 1, 2, \ldots, n$. 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 $\|x\|_2$. We denote the vector of ones of dimension m by e and the identity matrix of appropriate size by I. If f is a real valued function of the variable $x = (x_1, x_2, \ldots, x_n)^t \in R^n$ then the gradient of f is denoted by $\nabla f = (\partial f/\partial x_1, \ldots, \partial f/\partial x_n)^t$ and the Hessian matrix of f is denoted by $\nabla^2 f = (\partial^2 f/\partial x_i \partial x_j)_{i,j=1,2,\ldots,n}$.

The remainder of this paper is organized as follows. Section 2 reviews the TWSVM formulation. Our proposed formulation is presented and discuss the related theoretical analysis in Sect. 3. Numerical experiments have been performed on a number of interesting synthetic and real-world benchmark datasets and their results are compared with other SVMs in Sect. 4. Finally, concluding remarks and future work are given in Sect. 5.

2 Twin support vector machine for classification

In this section, we give a brief description of TWSVM formulation. For details, the interested reader is referred to [8, 14].

In 2007, Jayadeva et al. [14] proposed a new non-parallel support vector machine for binary classification, termed as TWSVM, which is in the spirit of GEPSVM. However, TWSVM has the formulation similar to typical support vector machine (SVM) formulation [6, 44] except that not all the patterns appear in the constraints of either problem at the same time. This makes TWSVM [14] faster than standard SVM.

Suppose that all the data points in class +1 are denoted by a matrix $A \in R^{m_1 \times n}$, where the ith row $A_i \in R^n$ and the matrix $B \in R^{m_2 \times n}$ represent the data points of class −1. Unlike SVM, the linear TWSVM [14] seeks a pair of non-parallel hyperplanes

$$f_1(x) = w_1^t x + b_1 \quad \text{and} \quad f_2(x) = w_2^t x + b_2. \tag{1}$$

The idea in the linear TWSVM is to solve the following two QPPs with objective functions corresponding to one of the two classes and constraints corresponding to the other class:

$$\begin{aligned}\min_{(w_1,b_1)\in R^{n+1}} \quad & \frac{1}{2}\|Aw_1+e_2b_1\|^2+C_1\|\xi_1\| \\ s.t. \quad & -(Bw_1+e_1b_1)+\xi_1\geq e_1, \quad \xi_1\geq 0,\end{aligned} \tag{2}$$

$$\begin{aligned}\min_{(w_2,b_2)\in R^{n+1}} \quad & \frac{1}{2}\|Bw_2+e_1b_2\|^2+C_2\|\xi_2\| \\ s.t. \quad & (Aw_2+e_2b_2)+\xi_2\geq e_2, \quad \xi_2\geq 0,\end{aligned} \tag{3}$$

where $C_1, C_2 > 0$ are parameters and e_1, e_2 are vectors of one of appropriate dimensions. It is evident that the idea in TWSVM is to solve two QPPs (2) and (3), each of the QPPs in the TWSVM pair is a typical SVM formulation, except that not all data points appear in the constraints of either problem [14].

In order to derive the corresponding dual formulation, TWSVM assumes that the matrices G^tG and H^tH, where $G=[A\ \ e_2]$ and $H=[B\ \ e_1]$, are non-singular. The dual QPPs are

$$\begin{aligned}\min_{u_1\in R^{m_2}} \quad & \frac{1}{2}u_1^tH(G^tG)^{-1}H^tu_1-e_1^tu_1 \\ s.t. \quad & 0\leq u_1\leq C_1,\end{aligned} \tag{4}$$

$$\begin{aligned}\min_{u_2\in R^{m_1}} \quad & \frac{1}{2}u_2^tG(H^tH)^{-1}G^tu_2-e_2^tu_2 \\ s.t. \quad & 0\leq u_2\leq C_2.\end{aligned} \tag{5}$$

Remark 1 The matrices G^tG and H^tH appearing in the dual formulation (4) and (5) may be singular. To avoid the possible ill conditioning, the inverse matrices $(G^tG)^{-1}$ and $(H^tH)^{-1}$ are approximately replaced by $(G^tG+\delta I)^{-1}$ and $(H^tH+\delta I)^{-1}$, where δ is a very small positive scalar and I is an identity matrix of appropriate dimensions.

Thus the nonparallel proximal hyperplanes are obtained from the solution u_1 and u_2 of (4) and (5) by

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = -(G^tG+\delta I)^{-1}H^tu_1 \quad \text{and} \quad \begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = (H^tH+\delta I)^{-1}G^tu_2. \tag{6}$$

The dual problems (4) and (5) are derived and solved in [14]. Experimental results of Jayadeva et. al. [14] show that the performance of TWSVM is better than the conventional SVM and GEPSVM on UCI machine learning datasets.

3 Proposed Newton method for implicit Lagrangian twin support vector machines (NLTSVM)

Motivated by the works of [11, 24, 37], we proposed in this paper an implicit Lagrangian formulation for the twin support vector machine (TWSVM) classifiers by formulating a pair of unconstrained minimization problems in dual variables whose solutions will be obtained using finite Newton method. The idea of our NLTSVM is to reformulate TWSVM as a strongly convex problem by incorporated regularization techniques to improve the training speed and robustness. 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 and TBSVM, which leads to extremely simple and fast algorithm.

3.1 Linear case

Different from TWSVM, the primal problems of TWSVM (2) and (3) are modified as follows:

$$\begin{aligned}\min_{(w_1,b_1)\in R^{n+1}} \quad & \frac{1}{2}\|Aw_1+e_2b_1\|^2+\frac{C_1}{2}\|\xi_1\|^2+\frac{C_3}{2}\left\|\begin{bmatrix} w_1 \\ b_1 \end{bmatrix}\right\|^2 \\ s.t. \quad & -(Bw_1+e_1b_1)+\xi_1\geq e_1,\end{aligned} \tag{7}$$

$$\begin{aligned}\min_{(w_2,b_2)\in R^{n+1}} \quad & \frac{1}{2}\|Bw_2+e_1b_2\|^2+\frac{C_2}{2}\|\xi_2\|^2+\frac{C_4}{2}\left\|\begin{bmatrix} w_2 \\ b_2 \end{bmatrix}\right\|^2 \\ s.t. \quad & (Aw_2+e_2b_2)+\xi_2\geq e_2.\end{aligned} \tag{8}$$

Note that there are three terms in the objective functions of (7) and (8). The first term in the objective function of (7) or (8) is the sum of squared distances from the hyperplane to points of one class. Therefore, minimizing it tends to keep the hyperplane close to points of one class (say positive class). The constraints require the hyperplane to be at a distance of atleast 1 from points of the other class (say negative class). The second term is the sum of error variables related with the constraints, requiring the distances from the two hyperplanes to points in the other class to be one or greater. The third term is a regularization term that is similar to [6, 23, 37, 40, 41] which make the objective functions strongly convex. Thus, it has a unique global optimal solution.

Our NLTSVM replaces the slack variables with 2-norm instead of 1-norm as in TWSVM and TBSVM, which make the constraints ($\xi_1, \xi_2 \geq 0$) redundant. 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 and TBSVM, which leads to extremely simple and fast algorithm. Similar to TBSVM, SRM principle is implemented in our NLTSVM by adding a regularization term with the idea of maximizing the margin and only empirical risk is minimized in the primal problems of TWSVM, STWSVM and LSTWSVM. Similar to standard SVM, this strategy leads our formulation to be more theoretically sound than the original TWSVM, STWSVM and LSTWSVM. The computational results, given in Sect. 4, clearly show that our NLTSVM does not compromise on generalization performance.

One can obtain the dual QPPs of (7) and (8) as follows:

$$\min_{0 \le u_1 \in R^{m_2}} L_1(u_1) = \frac{1}{2} {u_1}^t Q_1 u_1 - {e_1}^t u_1 \tag{9}$$

$$\min_{0 \le u_2 \in R^{m_1}} L_2(u_2) = \frac{1}{2} {u_2}^t Q_2 u_2 - {e_2}^t u_2 \tag{10}$$

where

$$Q_1 = \frac{I}{C_1} + N_1; \quad Q_2 = \frac{I}{C_2} + N_2; \quad N_1 = H(G^t G + C_3 I)^{-1} H^t;$$
$$N_2 = G(H^t H + C_4 I)^{-1} G^t; \quad G = [A\ e_2] \quad \text{and} \quad H = [B\ e_1].$$

The nonparallel proximal hyper planes are obtained from the solution u_1 and u_2 of (9) and (10) by

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = -(G^t G + C_3 I)^{-1} H^t u_1 \quad \text{and}$$
$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = (H^t H + C_4 I)^{-1} G^t u_2. \tag{11}$$

Remark 2 Unlike TWSVM, the matrices appearing in the dual objective functions (9) and (10) of our NLTSVM are positive definite and there is no upper bound on the dual variables u_1 and u_2.

Note that the regularization parameter δ used in TWSVM formulation is just a fixed small scalar while penalty parameters C_3, C_4 used in our NLTSVM are weighting factors which determine the trade-off between the regularization term and the empirical risk. Therefore, selecting appropriate parameters C_3, C_4 reflects the SRM principle. We have seen in the experimental section that the classification accuracy improved on adjusting the values of C_3, C_4.

3.2 Nonlinear case

In this subsection, we extend our NLTSVM to the nonlinear case using kernel trick. We consider the following kernel based surfaces instead of hyperplanes:

$$K(x^t, C^t) w_1 + b_1 = 0 \quad \text{and} \quad K(x^t, C^t) w_2 + b_2 = 0, \tag{12}$$

where $C^t = [A\ B]^t$ and K is appropriately chosen kernel.

Similar to linear case, the optimization problem for our NLTSVM in the kernel feature space can be reformulated as

$$\min_{(w_1,b_1) \in R^{m+1}} \frac{1}{2} \|K(A, C^t) w_1 + e_2 b_1\|^2 + \frac{C_1}{2} \|\xi_1\|^2 + \frac{C_3}{2} \left\| \begin{bmatrix} w_1 \\ b_1 \end{bmatrix} \right\|^2$$
$$\text{s.t.} \quad -(K(B, C^t) w_1 + e_1 b_1) + \xi_1 \ge e_1, \tag{13}$$

$$\min_{(w_2,b_2) \in R^{m+1}} \frac{1}{2} \|K(B, C^t) w_2 + e_1 b_2\|^2 + \frac{C_2}{2} \|\xi_2\|^2 + \frac{C_4}{2} \left\| \begin{bmatrix} w_2 \\ b_2 \end{bmatrix} \right\|^2$$
$$\text{s.t.} \quad (K(A, C^t) w_2 + e_2 b_2) + \xi_2 \ge e_2. \tag{14}$$

where $K(A, C^t)$ and $K(B, C^t)$ are kernel matrices of sizes $m_1 \times m$ and $m_2 \times m$ respectively, where $m = m_1 + m_2$. Similar to linear case, the nonparallel proximal hyper planes are obtained as

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = -(S^t S + C_3 I)^{-1} R^t u_1 \quad \text{and}$$
$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = (R^t R + C_4 I)^{-1} S^t u_2, \tag{15}$$

where $S = [K(A, C^t)\ e_2]$ and $R = [K(B, C^t)\ e_1]$.

3.3 Method of solution

In this subsection, we will discuss a fast and effective Newton algorithm for solving UMPs (9) and (10) by solving a system of linear equations in a finite number of times.

Our NLTSVM algorithm is based directly on applying the Karush–Kuhn–Tucker (KKT) necessary and sufficient optimality conditions [21, 24] for the dual problems (9) and (10):

$$0 \le u_1 \perp (Q_1 u_1 - e_1) \ge 0 \quad \text{and} \quad 0 \le u_2 \perp (Q_2 u_2 - e_2) \ge 0. \tag{16}$$

By using the well-known identity between two vectors (or real numbers) a and b:

$$0 \le a \perp b \ge 0 \quad \text{if and only if} \quad a = (a - \alpha b)_+$$
$$\text{for any} \quad \alpha \ge 0.$$

The solutions of the following equivalent pair of problems will be considered [24]: for any $\alpha_1, \alpha_2 > 0$,

$$(Q_1 u_1 - e_1) = (Q_1 u_1 - \alpha_1 u_1 - e_1)_+ \quad \text{and}$$
$$(Q_2 u_2 - e_2) = (Q_2 u_2 - \alpha_2 u_2 - e_2)_+. \tag{17}$$

The optimality condition (17) becomes necessary and sufficient condition to be satisfied by the unconstrained minimum of the following pair of implicit Lagrangian's [25] associated to the pair of dual problems (9) and (10): for any $\alpha_1, \alpha_2 > 0$,

$$\min_{u_1\in R^{m_2}} L_1(u_1) = \frac{1}{2}{u_1}^t Q_1 u_1 - {e_1}^t u_1 + \frac{1}{2\alpha_1}\left(\|(Q_1u_1-\alpha_1u_1-e_1)_+\|^2 - \|Q_1u_1-e_1\|^2\right) \tag{18}$$

$$\min_{u_2\in R^{m_1}} L_2(u_2) = \frac{1}{2}{u_2}^t Q_2 u_2 - {e_2}^t u_2 + \frac{1}{2\alpha_2}\left(\|(Q_2u_2-\alpha_2u_2-e_2)_+\|^2 - \|Q_2u_2-e_2\|^2\right) \tag{19}$$

where $\alpha_k, k=1,2$ is sufficiently large but finite positive parameter. The implicit Lagrangian formulations [11, 24, 36] consist of replacing the non-negativity constrained quadratic minimization problems (9) and (10) by the equivalent unconstrained piecewise quadratic minimization problems (18) and (19).

Our finite Newton method consists of applying Newton's method to UMPs (18) and (19) and showing that it terminates in a finite number of steps at the global minimum. The gradient of $L_k(u_k)$ is:

$$\nabla L_k(u_k) = \left(\frac{\alpha_kI-Q_k}{\alpha_k}\right)\left[(Q_ku_k-e)-(Q_ku_k-\alpha_ku_k-e)_+\right]. \tag{20}$$

Notice that for $k=1,2$, the gradient $\nabla L_k(u_k)$ is not differentiable and therefore the Hessian matrix of second order partial derivatives of $L_k(u_k)$ is not defined in the usual sense. The generalized Hessian of $L_k(u_k)$ in the sense of Hiriart-Urruty et al. [13] exists and is defined as follows:

$$\partial^2 L_k(u_k) = \left(\frac{\alpha_kI-Q_k}{\alpha_k}\right)\left[Q_k + \mathrm{diag}(Q_ku_k-\alpha_ku_k-e)_*(\alpha_kI-Q_k)\right], \tag{21}$$

where diag $(.)_*$ denote a diagonal function and $(.)_*$ denotes the step function. For $k=1,2$, the basic Newton step of the iterative algorithm is in determining the unknowns u_k^{i+1} at the $(i+1)^{th}$ iteration using the current i^{th} iterate u_k^i using

$$\nabla L_k(u_k^i) + \partial^2 L_k(u_k^i)(u_k^{i+1}-u_k^i) = 0 \quad \text{where} \quad i=0,1,2,\ldots. \tag{22}$$

Using the property that the matrices $Q_k, k=1,2$ defined in (9) and (10) are positive definite and choosing the parameter α_k satisfying the condition [11]: $\alpha_k > \|Q_k\|$ where $k=1,2$, the Newton's iterative step can be rewritten in the following simpler form: for $i=0,1,2,\ldots$

$$u_k^{i+1} = u_k^i - \partial h_k(u_k^i)^{-1} h_k(u_k^i). \tag{23}$$

For $k=1,2, h_k(u_k)$ and $\partial h_k(u_k)$ are defined as:

$$\begin{aligned} h_k(u_k) &:= (Q_ku_k-e)-(Q_ku_k-\alpha_ku_k-e)_+ \\ &= \left(\frac{\alpha_kI-Q_k}{\alpha_k}\right)^{-1}\nabla L(u_k), \\ \partial h_k(u_k) &:= Q_k + diag(Q_ku_k-\alpha_ku_k-e)_*(\alpha_kI-Q_k) \\ &= \left(\frac{\alpha_kI-Q_k}{\alpha_k}\right)^{-1}\partial^2 L(u_k). \end{aligned} \tag{24}$$

We will be used simpler iteration (24) in our implementation instead of the equivalent iteration (20). By defining the following matrix: for $k=1,2$,

$$E_k = \text{ diag } (Q_ku_k-\alpha_ku_k-e)_*, Z_k = {F_k}^{-1}(I-E_k) \quad \text{and} \quad F_k = \alpha E_k + \frac{I-E_k}{C_k}.$$

One can obtain

$$\partial h_k(u_k)^{-1} = (I+Z_kN_k)^{-1}{F_k}^{-1}. \tag{25}$$

Now we state Newton algorithm for solving unconstrained minimization problems (18) and (19) for an arbitrary positive definite matrix Q_k using the simplified iteration (23) together with an Armijo stepsize [1, 20] in order to guarantee finite termination from any starting point.

```
Newton algorithm with Armijo step size [11, 22]. For solving pair of UMPs (18) and(19) with $k = 1, 2$ :
Start with any initial guess $u_k^0$ and let $i = 0$
(i) Compute $\nabla L_k(u_k^i)$
(ii) Stop the iteration if $\nabla L_k(u_k^i) = 0$
Else
•  Compute $\nabla^2 L_k(u_k^i)$
•  Determine the direction vector $d_k^i \in R^m$ as the solution of the following linear system of equations in $m$ variables
       $\nabla^2 L_k(u_k^i) d_k^i = -\nabla L_k(u_k^i)$
(iii) Armijo step size: Define
       $u_k^{i+1} = u_k^i + \lambda_i d_k^i$
where $\lambda_i = \max\{1, \frac{1}{2}, \frac{1}{4}, \ldots\}$ is the step size in which
$L_k(u_k^i) - L_k(u_k^i + \lambda_i d_k^i) \geq -\delta\lambda_i \nabla L_k(u_k^i)^t d_k^i$, for some $\delta \in (0, \frac{1}{2})$
(iv) Replace $i$ by $i + 1$ and go to (i).
```

The convergence and finite termination of the above Newton algorithm with Armijo stepsize will follow as a simple extension of [22, 24].

Note that the computational complexity of SVM and TWSVM are $O(m^3)$ and $O(2\times(m/2)^3)$ respectively, where m is the total size of training data. It means that TWSVM is approximately four times faster than SVM. The linear NLTSVM solves just two matrix inversions with order of $(n+1)\times(n+1)$ where $n<<m$. For nonlinear NLTSVM, the inverses of the matrices with order of $(m+$

$1) \times (m+1)$ is required. We have been utilized Sherman–Morrison–Woodbury (SMW) formula to reduce the computational cost and need inverses of smaller dimension $(m_1 \times m_1)$ and $(m_2 \times m_2)$ to solve (15).

4 Experimental results

In order to evaluate the efficiency of our NLTSVM, numerical experiments were performed on 'Cross-Planes' and 'Ripley' datasets as examples of synthetic datasets and several well-known, publicly available, benchmark datasets, and their results were compared with GEPSVM, TWSVM, STWSVM and LSTWSVM. All the experiments were performed in MATLAB R2010a environment on a PC running on Windows XP OS with 3.30 GHz Intel (R) Core (TM) i3-2120 processor having 4 GB of RAM. In our experiment, we employ optimization toolbox of MATLAB for GEPSVM and TWSVM. In all the examples considered, the Gaussian 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 taken. The classification accuracy of each algorithm was computed using the well-known ten-fold cross-validation methodology [9]. For brevity's sake, we set $C_1 = C_2$ for TWSVM, STWSVM and LSTWSVM, $C_1 = C_2, C_3 = C_4$ for our NLTSVM, and the kernel parameter value μ were allowed to vary from the sets $\{10^{-5}, 10^{-4}, \ldots, 10^{5}\}$ and $\{2^{-10}, 2^{-9}, \ldots, 2^{10}\}$ respectively. For GEPSVM, the range of δ was allowed to vary from the set $\{2^{-7}, 2^{-6}, \ldots, 2^{7}\}$. Finally, choosing these optimal values, the classification accuracies and computational efficiencies on the test dataset was calculated.

The accuracy used to evaluate methods is defined as follows:

$$\text{Accuracy} = (TP + TN)/(TP + TN + FP + FN),$$

where TP, TN, FP and FN are the number of true positive, true negative, false positive and false negative respectively.

We conduct the experiments on five synthetic datasets and fifteen real world benchmark datasets to investigate the performance of our NLTSVM. The performance comparisons of five algorithms are summarized in Table 1, where "Accuracy" denotes the mean value of ten-testing results and "Time" denotes the mean value of the time taken by ten experiments. We draw some conclusions after implementing these experiments on twenty datasets. In terms of prediction accuracy, our proposed NLTSVM yields the highest accuracy among five algorithms on most of the datasets considered. The main reason of our proposed NLTSVM yields such good testing accuracy is that our NLTSVM solves two systems of linear equations instead of solving two QPPs. The next good algorithm is LSTWSVM which yields slightly lower testing accuracy than TWSVM, STWSVM and our NLTSVM, but higher than GEPSVM for most of the datasets. Among five algorithms, GEPSVM yields lowest testing accuracy for most of the datasets considered. Moreover on Heart-Statlog, Ionosphere, Bupa, Transfusion, Haberman, Sonar, Wpbc, Cleve, Monks2, Monks3 and Splice datasets, GEPSVM produces extremely low testing accuracy compared with four other algorithms. Table 1 shows the comparison of computational time and accuracy for all five algorithms with Gaussian kernel. It is evident that our NLTSVM have performed several orders of magnitude faster than GEPSVM and TWSVM but nearly similar to STWSVM and LSTWSVM. Our NLTSVM outperforms GEPSVM and TWSVM on most of the datasets considered which clearly indicates the overall superiority. Also it is worth mentioning that NLTSVM does not require any special optimizers.

4.1 Synthetic datasets

In this subsection, we consider five examples to compare our NLTSVM with the other SVMs. First we take a simple two dimensional "Cross Planes" dataset as an example of synthetic dataset which was also tested in [26, 37, 39–41]. It was generated by perturbing points lying on two intersecting lines and the intersection point is not in the center. The linear classifiers obtained by TWSVM and our NLTSVM along with the input data are shown in Fig. 1a, b respectively. In the figures, positive points are plotted as "+" and negative ones are plotted as "○". The corresponding hyperplanes has also been plotted in Fig. 1a, b. It is easy to see that the result of the proposed NLTSVM is more reasonable than that of TWSVM. This indicate that our NLTSVM can handle the "Cross Planes" dataset much better than TWSVM. The classification accuracy and central processing unit (CPU) time of each algorithm are summarized in Table 1. The results clearly demonstrate the superiority of multi-plane/surface classifiers over GEPSVM, TWSVM, STWSVM and LSTWSVM. The second example is an artificial-generated Ripley's synthetic dataset [35]. It is also two dimensional dataset which includes 250 training points and 1000 test points. Table 1 shows the learning results of nonlinear GEPSVM, TWSVM, STWSVM, LSTWSVM and NLTSVM. It can be seen that our NLTSVM obtains better classification accuracy with less training time than GEPSVM, TWSVM, STWSVM and LSTWSVM which indicate the suitability of our NLTSVM for these kind of problems since its non-parallel hyperplanes successfully describe the two class of points.

Table 1 Performance comparisons of GEPSVM, TWSVM, STWSVM, LSTWSVM and NLTSVM on synthetic and real world datasets using Gaussian kernel

Datasets (train size, test size)	GEPSVM Accuracy (%) Time (s)	TWSVM Accuracy (%) Time (s)	STWSVM Accuracy (%) Time (s)	LSTWSVM Accuracy (%) Time (s)	NLTSVM Accuracy (%) Time (s)
Cross planes	90.00	97.50	97.50	94.29	**100**
(80×2, 70×2)	0.1782	0.8575	0.010	0.0198	0.0333
Ripley	77.30	85.20	88.80	89.40	**90.60**
(1000×2, 250×2)	0.9065	0.1280	0.1253	0.1924	0.0959
Heart-Statlog	75.71	81.43	80.00	80.00	**85.71**
(200×13, 70×13)	0.8125	0.4404	0.0145	0.0797	0.0109
WDBC	84.06	78.26	84.06	**85.50**	84.06
(500×30, 69×30)	4.9375	0.6929	0.0065	0.0078	0.1014
Ionosphere	73.33	94.29	95.23	**96.19**	95.24
(246×34, 105×34)	1.4843	0.3374	0.0174	0.0162	0.0158
Bupa Liver	46.15	64.42	63.46	55.76	**65.38**
(241×6, 104×6)	1.125	0.1799	0.0186	0.0128	0.0143
Votes	93.02	**96.90**	95.35	96.12	95.35
(306×16, 129×16)	1.7320	0.3638	0.0046	0.0224	0.0279
WPBC	64.91	75.44	75.44	71.92	**77.19**
(137×33, 57×33)	0.3750	0.2142	0.0051	0.0044	0.0042
Cleve	71.67	72.50	75.26	80.00	**81.67**
(177×13, 120×13)	0.5625	0.2655	0.009	0.0299	0.0070
Monk-2	78.94	87.27	90.56	85.64	**96.30**
(432×7, 122×7)	0.6400	0.0984	0.0245	0.0305	0.0066
Monk-3	81.48	90.74	85.53	93.51	**94.68**
(432×7, 122×7)	0.6875	0.1773	0.0046	0.0269	0.0032
Australian	89.33	76.00	76.00	78.00	**90.67**
(540×14, 150×14)	5.7812	1.6721	0.1426	0.1522	0.1229
Transfusion	86.49	**90.54**	**90.54**	**90.54**	**90.54**
(600×4, 148×4)	6.7812	0.2094	0.1234	0.1000	0.1626
Haberman	66.98	76.41	76.41	75.47	**77.36**
(200×3, 106×3)	1.078	1.2963	0.0102	0.0344	0.0095
Sonar	70.69	77.58	75.68	**82.75**	79.31
(150×60, 58×60)	0.500	0.0822	0.0040	0.0242	0.0052
Splice	67.03	**88.33**	88.03	88.03	85.76
(500×60, 2675×60)	673.359	0.6190	0.1348	0.1207	0.1013
Tic-Tac-Toe	**94.43**	**94.43**	**94.43**	91.28	**94.43**
(671×9, 287×9)	1.7892	7.15085	0.1204	0.2961	0.6547

Bold values indicate the best result

To further show the advantage of our NLTSVM in the training speed, we have compared the computational time of our NLTSVM with GEPSVM, TWSVM, STWSVM and LSTWSVM on three NDC [28] datasets. The parameters of all the algorithms have been fixed i.e. $C_{1,2,3,4} = 1$ and $\mu = 2^{-8}$. We have selected 10 % data for testing and rest for training. The central processing unit (CPU) time of each algorithm are summarized in Table 2. One can see from the Table 2 that our NLTSVM is more reasonable than GEPSVM, TWSVM, STWSVM and LSTWSVM.

4.2 UCI datasets

To further evaluate the classification ability of our NLTSVM, we compare the behavior of our NLTSVM with GEPSVM, TWSVM, STWSVM and LSTWSVM on several publicly available benchmark datasets [27]. In all the

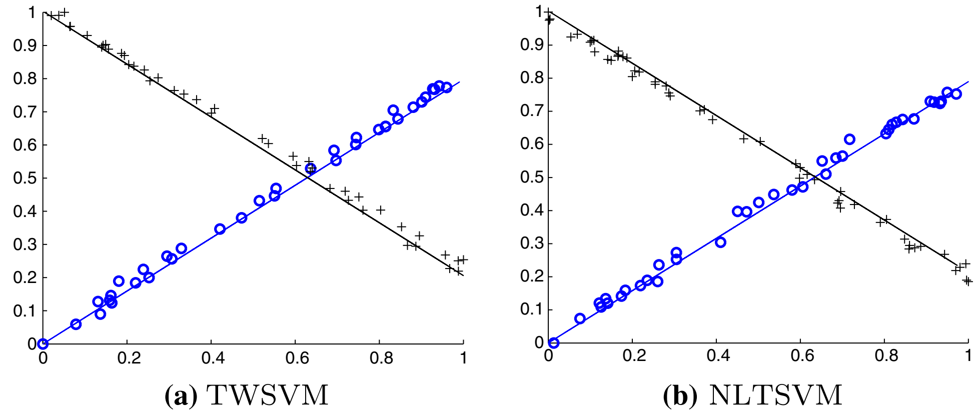


Fig. 1 Classification results of **a** TWSVM **b** NLTSVM for Cross planes dataset

Table 2 Comparison on NDC datasets with Gaussian kernel

Datasets	GEPSVM Time (s)	TWSVM Time (s)	STWSVM Time (s)	LSTWSVM Time (s)	NLTSVM Time (s)
NDC1k	34.7429	2.0138	0.7521	1.1822	0.9873
NDC3k	250.2833	5.625	3.2012	3.2177	3.1023
NDC5k	1093.80	10.4802	8.4696	11.3433	8.2302

real-world examples considered, each attribute of the original data is normalized as follows:

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

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 column of A.

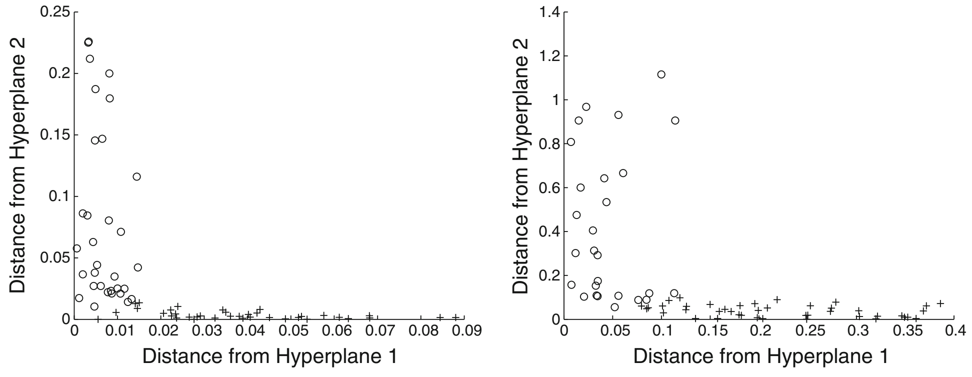
The sizes of training and test data, the number of attributes, training time and accuracies of each algorithm for nonlinear classifiers were summarized in Table 1 and the best accuracy is shown by bold figures. Clearly One can observe from Table 1 that, in comparison to GEPSVM, TWSVM, STWSVM and LSTWSVM, our NLTSVM shows better generalization performance. In details, for Heart-Statlog dataset, the experimental result (accuracy 85.71 %, time 0.0109 s) by our NLTSVM is higher than other four algorithms in computational time and classification accuracy, i.e., GEPSVM (accuracy 75.71 %, 0.8125 s), TWSVM (accuracy 81.43 %, 0.4404 s), STWSVM (accuracy 80.00 %, 0.0145 s) and LSTWSVM (accuracy 80.00 %, 0.0797 s). We obtained the similar conclusions for Bupa, WPBC, Cleve, Monks-2, Monks-3, Australian and Haberman datasets. For Votes dataset, the classification accuracy obtained by our NLTSVM (accuracy 95.35 %) is slightly lower than TWSVM (accuracy 96.90 %) and LSTWSVM (accuracy 96.12 %), it is higher than GEPSVM (accuracy 93.02 %). The empirical results further reveal that our NLTSVM, whose solutions are obtained by solving system of linear equations, is faster than TWSVM on all the datasets. It is worthwhile notice that choosing the values of the parameters C_3 and C_4 affect the results significantly and these values are varying in our NLTSVM rather than small fixed positive scalar in TWSVM. The details of optimal parameters for Gaussian kernel are listed in Table 3. It clearly indicates that adding the regularization terms in our formulation are useful.

To further compare our NLTSVM with TWSVM, we also compare it with the two-dimensional scatter plots that were obtained from the part test points for the WDBC and Heart Statlog datasets. The plots were obtained by plotting points with coordinates: perpendicular distance of a test point x from positive hyperplane 1 and the distance from negative hyperplane 2. In the figures, positive points are plotted as "+" and negative points are plotted as "∘". Hence, the clusters of points indicate how well the classification criterion is able to discriminate between the two classes. From Figs. 2a, b and 3a, b, it can be seen that our NLTSVM obtained large distances from the test samples to the opposite hyperplanes. In contrast, the TWSVM obtained small distances from the test points to the hyperplane pair. It means that our NLTSVM is much more robust when compared with the TWSVM.

For further fair comparisons, we use statistical test to demonstrate a correct analysis when comparing the performance of multiple algorithms over multiple datasets, which has been largely discussed. Since the Friedman test with the corresponding post hoc tests is pointed out to be a simple, safe, and robust non parametric test for comparison of more classifiers over multiple datasets [7], we use it to

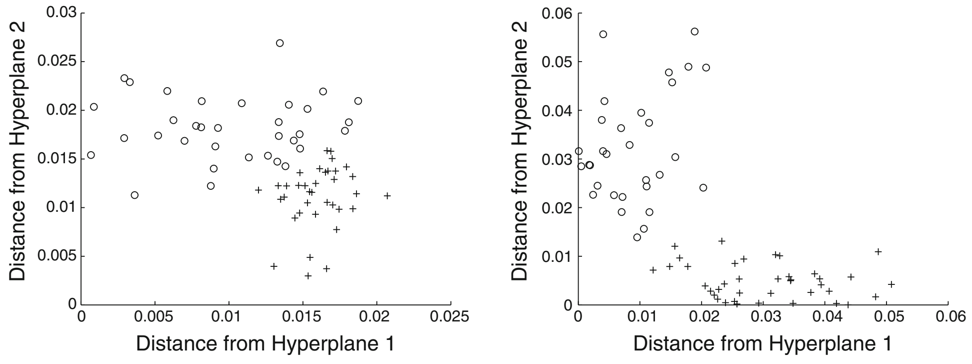
Table 3 Optimal parameters of GEPSVM, TWSVM, STWSVM, LSTWSVM and NLTSVM for Gaussian kernel

Datasets	GEPSVM (μ, δ)	TWSVM $(\mu, C_1 = C_2)$	STWSVM $(\mu, C_1 = C_2)$	LSTWSVM $(\mu, C_1 = C_2)$	NLTSVM $(\mu, C_1 = C_2, C_3 = C_4)$
Cross planes	$(2^9, 2^{-5})$	$(2^9, 10^{-5})$	$(2^8, 10^{-4})$	$(2^{-6}, 10^{-4})$	$(2^{-8}, 10^{-3}, 10^{-2})$
Ripley	$(2^{-5}, 2^4)$	$(2^7, 10^{-5})$	$(2^{-6}, 10^{-5})$	$(2^0, 10^{-1})$	$(2^{-8}, 10^{-3}, 10^{-2})$
Heart-Statlog	$(2^5, 2^2)$	$(2^{-5}, 10^{-5})$	$(2^5, 10^{-4})$	$(2^4, 10^{-1})$	$(2^{-5}, 10^4, 10^{-1})$
WDBC	$(2^4, 2^6)$	$(2^{-1}, 10^{-5})$	$(2^{-6}, 10^{-2})$	$(2^{-2}, 10^{-1})$	$(2^{-9}, 10^{-5}, 10^{-3})$
Ionosphere	$(2^3, 2^{-7})$	$(2^1, 10^{-3})$	$(2^2, 10^{-3})$	$(2^{-3}, 10^{-3})$	$(2^0, 10^{-4}, 10^{-1})$
Bupa	$(2^{-6}, 2^5)$	$(2^1, 10^{-5})$	$(2^4, 10^{-4})$	$(2^2, 10^{-1})$	$(2^{-1}, 10^{-5}, 10^{-2})$
Votes	$(2^{-4}, 2^{-1})$	$(2^{-7}, 10^{-5})$	$(2^{-6}, 10^{-4})$	$(2^2, 10^0)$	$(2^{-9}, 10^2, 10^{-3})$
WPBC	$(2^3, 2^6)$	$(2^0, 10^{-5})$	$(2^3, 10^{-5})$	$(2^4, 10^{-1})$	$(2^1, 10^{-5}, 10^{-1})$
Cleve	$(2^6, 2^2)$	$(2^{-5}, 10^{-5})$	$(2^9, 10^{-4})$	$(2^8, 10^{-4})$	$(2^{-9}, 10^5, 10^{-4})$
Monk-2	$(2^6, 2^{-3})$	$(2^1, 10^{-5})$	$(2^5, 10^4)$	$(2^{-2}, 10^{-1})$	$(2^{-4}, 10^{-3}, 10^{-3})$
Monk-3	$(2^6, 2^{-3})$	$(2^{-3}, 10^{-5})$	$(2^{-10}, 10^{-4})$	$(2^2, 10^{-1})$	$(2^{-3}, 10^{-5}, 10^{-4})$
Australian	$(2^0, 2^{-7})$	$(2^{-4}, 10^{-5})$	$(2^{-4}, 10^2)$	$(2^1, 10^{-1})$	$(2^{-2}, 10^5, 10^0)$
Transfusion	$(2^6, 2^{-3})$	$(2^{-8}, 10^{-1})$	$(2^{-10}, 10^1)$	$(2^1, 10^0)$	$(2^{-6}, 10^{-5}, 10^0)$
Haberman	$(2^6, 2^2)$	$(2^1, 10^{-5})$	$(2^6, 10^{-4})$	$(2^{-1}, 10^0)$	$(2^{-9}, 10^{-5}, 10^{-5})$
Sonar	$(2^1, 2^7)$	$(2^{-2}, 10^{-5})$	$(2^5, 10^3)$	$(2^0, 10^{-2})$	$(2^{-3}, 10^{-5}, 10^{-2})$
Splice	$(2^{-8}, 2^2)$	$(2^{-6}, 10^{-5})$	$(2^7, 10^{-4})$	$(2^2, 10^{-2})$	$(2^{-5}, 10^{-5}, 10^{-2})$
Tic-Tac-Toe	$(2^3, 2^2)$	$(2^{-6}, 10^{-5})$	$(2^{-8}, 10^{-2})$	$(2^1, 10^{-1})$	$(2^{-9}, 10^2, 10^{-5})$



(a) TWSVM **(b)** NLTSVM

Fig. 2 2-D projections of **a** TWSVM, **b** NLTSVM from WDBC dataset. *Plus* scatter plot of the positive points. *Circle* scatter plot of the negative points



(a) TWSVM **(b)** NLTSVM

Fig. 3 2-D projections of **a** TWSVM, **b** NLTSVM from Heart-Statlog dataset. *Plus* scatter plot of the positive points. *Circle* scatter plot of the negative points

Table 4 Average ranks of GEPSVM, TWSVM, STWSVM, LSTWSVM and NLTSVM with Gaussian kernel

Datasets	GEPSVM	TWSVM	STWSVM	LSTWSVM	NLTSVM
Cross planes	5	2.5	2.5	4	1
Ripley	5	4	3	2	1
Heart-Statlog	5	2	3.5	3.5	1
WDBC	3	5	3	1	3
Ionosphere	5	4	3	1	2
Bupa	5	2	3	4	1
Votes	5	1	3.5	2	3.5
WPBC	5	2.5	2.5	4	1
Cleve	5	4	3	2	1
Monk-2	5	3	2	4	1
Monk-3	5	3	4	2	1
Australian	2	4.5	4.5	3	1
Transfusion	5	2.5	2.5	2.5	2.5
Haberman	5	2.5	2.5	4	1
Sonar	5	3	4	1	2
Splice	5	1	2.5	2.5	4
Tic-Tac-Toe	2.5	2.5	2.5	5	2.5
Average Rank	4.56	2.88	3.03	2.79	1.74

compare the generalization ability of five algorithms. The average ranks of all the algorithms on accuracies were computed and listed in Table 4. We employ the Friedman test to check whether the measured average ranks are significantly different from the mean rank $R_j = 3$ expected under the null hypothesis:

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_{j=1}^{3} R_j^2 - \frac{k(k+1)^2}{4}\right]$$

is distributed according to χ_F^2 with $k-1$ degree of freedom. where k is the number of methods and N is the number of datasets.

$$\chi_F^2 = \frac{12\times 17}{5(5+1)} \times \left[4.56^2 + 2.88^2 + 3.03^2 + 2.79^2 + 1.74^2 - \frac{5(6)^2}{4}\right] = 27.7481.$$

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2} = \frac{(17-1)\times 27.7481}{17(5-1)-27.7481} = 11.0298.$$

With five algorithms and seventeen datasets, F_F is distributed according to the F-distribution with $(k-1)$ and $(k-1)(N-1) = (4,64)$ degrees of freedom. The critical value of $F(4,64)$ for $\alpha = 0.05$ is 2.52. So, we reject the null hypothesis $(F_F > F(4,64))$. We use the Nemenyi test for further pairwise comparison. According to [7], at $p = 0.10$, critical difference (CD) $= q_\alpha\sqrt{\frac{k(k+1)}{6N}} = 2.459\sqrt{\frac{5\times 6}{6\times 17}} = 1.336$. Since the difference between GEPSVM and our NLTSVM is larger than the critical difference $1.336(4.56 - 1.74 = 2.82 > 1.336)$, we can identify that the performance of NLTSVM is significantly better than GEPSVM. Similarly, we can identify that the performances of TWSVM, STWSVM and LSTWSVM are significantly better than GEPSVM. Further, we see that the difference between TWSVM, STWSVM, LSTWSVM and our NLTSVM is just below the critical difference, we can conclude that the post hoc test is not powerful enough to detect any significant difference between the algorithms. But the average rank of our NLTSVM is always lower than that of GEPSVM, TWSVM, STWSVM and LSTWSVM, once can conclude that our NLTSVM is better than GEPSVM, TWSVM, STWSVM and LSTWSVM.

Regarding the computational time of five algorithms shown in Table 1 reveals that the proposed NLTSVM is nearly seven times faster than GEPSVM and TWSVM. However, the proposed NLTSVM and GEPSVM cost nearly the same time. One of the reasons is that both are solving the systems of linear equations.

5 Conclusions and future work

In this paper, we proposed an implicit Lagrangian twin support vector machine (TWSVM) classifiers by formulating a pair of unconstrained minimization problems in dual variables whose solutions will be obtained using finite Newton method. The idea of our formulation is to reformulate TWSVM as a strongly convex problem by incorporated regularization techniques to improve the training speed and robustness. 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 and TBSVM, which leads to extremely simple and fast algorithm. To demonstrate the effectiveness of the proposed method, we performed numerical experiments on number of interesting real-world datasets and compared their results with other SVMs. Comparison of results with GEPSVM, TWSVM, STWSVM and LSTWSVM clearly demonstrate the effectiveness and suitability of the proposed method. Similar to TBSVM, there are four parameters in our NLTSVM, so the parameter selection is a practical problem and should be address in the future. Our future work will be on the extension of the proposed method to learning using privileged information.

Acknowledgments The author acknowledges the valuable comments of the anonymous reviewers and the Editor of Journal of Machine Learning and Cybernetics whose enthusiasm is gladly appreciated. Also, the author would like to express his sincere gratitude to Prof. Yuh-Jye Lee, NTUST Taiwan and Prof. S. Balasundaram, JNU New Delhi, India for their help during the preparation of this manuscript.

References

1. Armijo L (1966) Minimization of functions having Lipschitz-continuous first partial derivatives. Pac J Math 16:1–3
2. Balasundaram S, Tanveer M (2012) On proximal bilateral-weighted fuzzy support vector machine classifiers. Int J Adv Intell Paradig 4(3/4):199–210
3. Balasundaram S, Tanveer M (2013) On Lagrangian twin support vector regression. Neural Comput Appl 22(1):257–267
4. Balasundaram S, Tanveer M (2013) Smooth Newton method for implicit Lagrangian twin support vector regression. KES J 17(4):267–278
5. Brown MPS, Grundy WN, Lin D (2000) Knowledge-based analysis of micro-array gene expression data using support vector machine. Proc Natl Acad Sci USA 97(1):262–267
6. Cortes C, Vapnik VN (1995) Support vector networks. Mach Learn 20:273–297
7. Demsar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
8. Deng NY, Tian YJ, Zhang CH (2013) Support vector machines: optimization based theory, algorithms, and extensions. CRC Press, Boca Raton
9. Duda RO, Hart PR, Stork DG (2001) Pattern classification, 2nd edn. Wiley, New York
10. Fung G, Mangasarian OL (2001) Proximal support vector machines. In: Provost F, Srikant R, (eds) Proceedings KDD-2001: knowledge discovery and data mining, August 26–29, 2001, San Francisco, CA, New York, pp 76–86
11. Fung G, Mangasarian OL (2003) Finite Newton method for Lagrangian support vector machine classification. Neurocomputing 55(1–2):39–55
12. Gao S, Ye Q, Ye N (2011) 1-Norm least squares twin support vector machines. Neurocomputing 74:3590–3597
13. Hiriart-Urruty JB, Strodiot JJ, Nguyen VH (1984) Generalized Hessian matrix and second order optimality conditions for problems with CL1 data. Appl Math Optim 11:43–56
14. Jayadeva Khemchandani R, Chandra S (2007) Twin support vector machines for pattern classification. IEEE Trans Pattern Anal Mach Intell 29(5):905–910
15. Joachims T (1999) Making large-scale support vector machine learning practical. In: Advances in kernel methods. In: Support vector learning. MIT Press, Cambridge
16. Joachims T, Ndellec C, Rouveriol C (1998) Text categorization with support vector machines: learning with many relevant features. Eur Conf Mach Learn Chemnitz Ger 10:137–142
17. Kumar MA, Gopal M (2008) Application of smoothing technique on twin support vector machines. Pattern Recogn Lett 29:1842–1848
18. Kumar MA, Gopal M (2009) Least squares twin support vector machines for pattern classification. Expert Syst Appl 36:7535–7543
19. Lee YJ, Mangasarian OL (2001a) RSVM: reduced support vector machines. In: Proceedings of the first SIAM international conference on data mining, pp 5–7
20. Lee YJ, Mangasarian OL (2001b) SSVM: a Smooth support vector machine for classification. Comput Optim Appl 20(1):5–22
21. Mangasarian OL (1994) Nonlinear programming. SIAM, Philadelphia
22. Mangasarian OL (2002) A finite Newton method for classification. Optim Methods Softw 17:913–929
23. Mangasarian OL, Musicant DR (1999) Successive overrelaxation for support vector machines. IEEE Trans Neural Netw 10:1032–1037
24. Mangasarian OL, Musicant DR (2001) Lagrangian support vector machines. J Mach Learn Res 1:161–177
25. Mangasarian OL, Solodov MV (1993) Nonlinear complementarity as unconstrained and constrained minimization. Math Program Ser B 62:277–297
26. Mangasarian OL, Wild EW (2006) Multisurface proximal support vector classification via generalized eigenvalues. IEEE Trans Pattern Anal Mach Intell 28(1):69–74
27. Murphy PM, Aha DW (1992) UCI repository of machine learning databases. University of California, Irvine. http://www.ics.uci.edu/ mlearn
28. Musicant D (1998) NDC: normally distributed clustered datasets. Computer Sciences Department, University of Wisconsin, Madison. www.cs.wisc.edu/dmi/svm/ndc
29. Nasiri JA, Charkari NM, Jalili S (2014) Least squares twin multi-class classification support vector machine. Pattern Recogn. doi:10.1016/j.patcog.2014.09.020
30. Osuna E, Freund R, Girosi F (1997) Training support vector machines: an application to face detection. In: Proceedings of computer vision and pattern recognition, pp 130–136
31. Peng X (2010) TSVR: an efficient twin support vector machine for regression. Neural Netw 23(3):365–372
32. Peng X, Xu D (2013) Robust minimum class variance twin support vector machine classifier. Neural Comput Appl 22(5):999–1011
33. Platt J (1999) Fast training of support vector machines using sequential minimal optimization. In: Scholkopf B, Burges CJC, Smola AJ (eds) Advances in kernel methods-support vector learning. MIT Press, Cambridge, pp 185–208
34. Qi Z, Tian Y, Shi Y (2012) Laplacian twin support vector machine for semi-supervised classification. Neural Netw 35:46–53
35. Ripley BD (2008) Pattern recognition and neural networks. Cambridge University Press, Cambridge
36. Scholkopf B, Smola A (2002) Learning with kernels. MIT Press, Cambridge
37. Shao Y-H, Zhang CH, Wang XB, Deng NY (2011) Improvements on twin support vector machines. IEEE Trans Neural Netw 22(6):962–968

38. Shao YH, Deng NY, Yang ZM, Chen WJ, Wang Z (2012) Probabilistic outputs for twin support vector machines. Knowl-Based Syst 33:145–151
39. Shao YH, Chen WJ, Deng NY (2014) Nonparallel hyperplane support vector machine for binary classification problems. Inf Sci 263:22–35
40. Tanveer M (2014) Application of smoothing techniques for linear programming twin support vector machines. Knowl Inf Syst. doi:10.1007/s10115-014-0786-3
41. Tanveer M (2015) Robust and sparse linear programming twin support vector machines. Cogn Comput 7(1):137–149
42. Tian Y, Ping Y (2014) Large-scale linear nonparallel support vector machine solver. Neural Netw 50:166–174
43. Tsang IW, Kwok JT, Cheung PM (2005) Core vector machines: fast SVM training on very large datasets. J Mach Learn Res 6:363–392
44. Vapnik VN (2000) The nature of statistical learning theory, 2nd edn. Springer, New York
45. Xu Y, Wang L (2012) A weighted twin support vector regression. Knowl-Based Syst 33:92–101
46. Zhong P, Xu Y, Zhao Y (2012) Training twin support vector regression via linear programming. Neural Comput Appl 21(2):399–407