AIMS Mathematics, 10(5): 10413–10430.
DOI: 10.3934/math.2025474
Received: 26 February 2025
Revised: 23 April 2025
Accepted: 29 April 2025
Published: 06 May 2025

https://www.aimspress.com/journal/Math

Research article

Accelerated least squares twin support vector machine with L_1-norm regularization by ADMM

Rujira Fongmun[1] **and Thanasak Mouktonglang**[1,2,3,*]

[1] Department of Mathematics, Faculty of Science, Chiang Mai University, Chiang Mai 50200, Thailand

[2] Data Science Research Center, Faculty of Science, Chiang Mai University, Chiang Mai 50200, Thailand

[3] Advanced Research Center for Computational Simulation, Chiang Mai University, Chiang Mai 50200, Thailand

* **Correspondence:** Email: thanasak.m@cmu.ac.th.

Abstract: This paper introduces a modified least squares twin support vector machine (LSTSVM) designed to enhance classification accuracy and robustness in the presence of outliers and noisy datasets. Building on the traditional twin SVM (TWSVM) and LSTSVM frameworks, we propose replacing the L_2-norm of error variables with the L_1-norm to mitigate the influence of extreme values and improve sparsity in the solution. To address the computational challenges of large-scale datasets, we employ the alternating direction method of multipliers (ADMM) to efficiently decompose the optimization problem into smaller subproblems, ensuring scalability and reduced computational costs. Acceleration steps with a guard condition are also integrated to speed up convergence. Experimental evaluations demonstrate the proposed method's superior performance in terms of computational efficiency and classification accuracy compared to TWSVM and traditional LSTSVM, making it a promising solution for real-world applications in classification tasks involving noisy or imbalanced data.

Keywords: LSTSVM; ADMM; computational time; machine learning algorithms
Mathematics Subject Classification: 65K05, 90C20

1. Introduction

Support vector machines (SVMs) are robust computational tools for supervised learning, commonly employed in classification and regression tasks. With foundations in statistical learning theory and Bayesian principles, SVMs aim to identify an optimal separating hyperplane that maximizes the margin

between positive and negative examples [1]. This approach has proven effective in diverse applications such as particle identification, face recognition, text categorization, and bioinformatics.

Expanding on the conventional SVM framework, Mangasarian and Wild [2] introduced the generalized eigenvalue proximal support vector machine (GEPSVM), which addresses binary classification by identifying two distinct hyperplanes, one for each category. The closest hyperplane is assigned to data points based on proximity, transforming the problem into a generalized eigenvalue formulation. The solutions are obtained using eigenvectors associated with the smallest eigenvalues.

To further enhance classification efficiency, Jayadeva et al. [3] proposed the twin support vector machine (TWSVM), which also constructs two non-parallel hyperplanes. Unlike GEPSVM, TWSVM involves solving two smaller quadratic programming problems (QPPs) instead of a single large-scale QPP. This design reduces computational complexity, as shown in experimental evaluations that demonstrate TWSVM's superior performance over GEPSVM and standard SVMs on datasets from the University of California, Irvine (UCI) machine learning repository [4].

Focusing on computational simplicity and scalability, Kumar and Gopal [5] developed the least squares twin support vector machine (LSTSVM) as an extension of TWSVM [6]. LSTSVM reformulates the primal QPPs of TWSVM using least squares principles, replacing inequality constraints with equality constraints. Consequently, the optimization reduces to solving two systems of linear equations, bypassing the need for external optimizers. This approach effectively accommodates nonlinear kernels while maintaining computational efficiency. Empirical comparisons across various UCI and artificial datasets confirm LSTSVM's faster training time and competitive classification accuracy relative to TWSVM and traditional LSTSVM.

Despite these advantages, traditional LSTSVM minimizes the L_2-norm of error variables, making it sensitive to outliers. This sensitivity can degrade classification accuracy, especially in noisy or imbalanced datasets. To address this, several enhancements have been proposed to improve the robustness of LSTSVM and related models. Furthermore, the standard LSTSVM reduces the coefficients of irrelevant features without eliminating any of them entirely. As a result, if the dataset contains many irrelevant features, using the standard LSTSVM may lead to a complex model with numerous included features since none of the irrelevant coefficients are reduced to zero.

Gao et al. [7] introduced the L_1-norm least squares twin support vector machine, which replaces the L_2-norm with the L_1-norm in the objective function to promote robustness and handle outliers effectively. This reformulation also promotes sparsity and feature suppression. By converting the constrained problem into an unconstrained convex quadratic form, they solve it efficiently using a generalized Newton method.

Yan et al. [8] proposed the L_1-norm-based least squares twin bounded support vector machine, replacing all conventional L_2-norms with L_1-norms to reduce outlier influence. The optimization problems are addressed through an iterative reweighting technique.

Wang et al. [9] introduced the robust capped L_1-norm twin support vector machine with privileged information, which incorporates the learning using privileged information framework. The capped L_1-norm enhances robustness, while upper and lower bound constraints on both main and privileged features control noise sensitivity. An alternating minimization approach is used to solve the optimization problems.

In a related line of work, the robust capped L_1-norm projection twin support vector machine (CPTSVM) was proposed to improve the outlier resistance of PTSVM models by Yang et al. [10]. By

replacing the squared L_2-norm with the capped L_1-norm, the CPTSVM formulation increases classifier robustness in the presence of noise and outliers. Though the resulting problems are non-convex and non-smooth, an iterative algorithm with proven convergence is employed to solve them. Experiments on artificial and benchmark datasets demonstrate the model's robustness and effectiveness.

These related works inform the design of our proposed method, which integrates L_1-norm regularization into the LSTSVM framework and leverages the alternating direction method of multipliers (ADMM) [11–13] for scalable optimization. In contrast to prior methods, our approach introduces acceleration mechanisms and guard conditions to ensure both robustness and fast convergence on large and noisy datasets.

2. Preliminaries

This section briefly reviews the fundamental concepts of TWSVM, LSTSVM, ADMM, and the Lasso technique.

2.1. Twin support vector machine (TWSVM)

TWSVM is a classification technique developed to reduce the computational burden of traditional SVM. Instead of finding a single hyperplane to separate two classes, TWSVM constructs two non-parallel hyperplanes. Each hyperplane is positioned closer to one class while maintaining the maximum possible distance from the other. Given a dataset D with m_1 and m_2 training points labeled $+1$ and -1, respectively, in $\mathbb{R}^n$, the data points for class $+1$ are represented by matrix $A \in \mathbb{R}^{m_1 \times n}$, while matrix $B \in \mathbb{R}^{m_2 \times n}$ represents class -1. The linear TWSVM is defined by:

$$x^T w_1 + b_1 = 0$$

and

$$x^T w_2 + b_2 = 0,$$

where $w_1, w_2 \in \mathbb{R}^n$ are normal vectors, and $b_1, b_2 \in \mathbb{R}$ are bias terms. These hyperplanes are obtained by solving two separate optimization problems, each associated with one class:

$$\begin{array}{ll} \min\limits_{w_1, b_1, \xi_2} & \dfrac{1}{2}\|Aw_1 + e_1 b_1\|_2^2 + c_1 e_2^T \xi_2 \\ \text{s.t.} & -(Bw_1 + e_2 b_1) + \xi_2 \geq e_2, \\ & \xi_2 \geq 0, \end{array} \tag{2.1}$$

and

$$\begin{array}{ll} \min\limits_{w_2, b_2, \xi_1} & \dfrac{1}{2}\|Bw_2 + e_2 b_2\|_2^2 + c_2 e_1^T \xi_1 \\ \text{s.t.} & (Aw_2 + e_1 b_2) + \xi_1 \geq e_1, \\ & \xi_1 \geq 0, \end{array} \tag{2.2}$$

where $c_1, c_2 > 0$ are parameters, $\xi_1 \in \mathbb{R}^{m_1}$, $\xi_2 \in \mathbb{R}^{m_2}$ are slack vectors, and e_1 and e_2 are vectors of ones of appropriate dimensions.

Using the Lagrangian dual method on (2.1) and (2.2), and introducing the Lagrange multipliers $\alpha \in \mathbb{R}^{m_2}$ and $\beta \in \mathbb{R}^{m_1}$, the resulting Wolfe dual formulations are:

$$\begin{array}{ll} \max\limits_{\alpha} & e_2^T\alpha - \dfrac{1}{2}\alpha^T G\left(H^T H\right)^{-1} G^T\alpha \\ \text{s.t.} & 0 \le \alpha \le c_1, \end{array} \tag{2.3}$$

where

$$G = \begin{bmatrix} B & e_2 \end{bmatrix}$$

and

$$H = \begin{bmatrix} A & e_1 \end{bmatrix}.$$

And

$$\begin{array}{ll} \max\limits_{\beta} & e_1^T\beta - \dfrac{1}{2}\beta^T P\left(Q^T Q\right)^{-1} P^T\beta \\ \text{s.t.} & 0 \le \beta \le c_2, \end{array} \tag{2.4}$$

where

$$P = \begin{bmatrix} A & e_1 \end{bmatrix}$$

and

$$Q = \begin{bmatrix} B & e_2 \end{bmatrix}.$$

Solving (2.3) and (2.4) yields the two non-parallel hyperplanes:

$$\begin{array}{ll} u = & -\left(H^T H\right)^{-1} G^T\alpha, \text{ where } u = \begin{bmatrix} w_1 & b_1 \end{bmatrix}^T, \\ v = & \left(Q^T Q\right)^{-1} P^T\beta, \text{ where } v = \begin{bmatrix} w_2 & b_2 \end{bmatrix}^T. \end{array} \tag{2.5}$$

In comparison to SVM, the QPP in TWSVM has fewer parameters than that of SVM since (2.3) and (2.4) each involve only m_1 and m_2 parameters, unlike SVM's QPP, which depends on $m_1 + m_2$ parameters.

In cases involving nonlinear data, the kernel functions are introduced, allowing the two hyperplanes of TWSVM in the kernel space to be represented as follows:

$$K\left(x^T, C^T\right)u_1 + b_1 = 0 \quad \text{and} \quad K\left(x^T, C^T\right)u_2 + b_2 = 0, \tag{2.6}$$

where K is a properly selected kernel,

$$C = \begin{bmatrix} A & B \end{bmatrix}^T$$

and $u_1, u_2 \in \mathbb{R}^n$. The nonlinear TWSVM optimization problem can be expressed:

$$\begin{array}{ll} \min\limits_{w_1,b_1,\xi_2} & \dfrac{1}{2}\|K\left(A, C^T\right)u_1 + e_1 b_1\|_2^2 + c_1 e_2^T\xi_2 \\ \text{s.t.} & -\left(K\left(B, C^T\right)u_1 + e_2 b_1\right) + \xi_2 \ge e_2, \\ & \xi_2 \ge 0, \end{array} \tag{2.7}$$

and

$$\begin{array}{ll} \min\limits_{w_2,b_2,\xi_1} & \dfrac{1}{2}\|K\left(B, C^T\right)u_2 + e_2 b_2\|_2^2 + c_2 e_1^T\xi_1 \\ \text{s.t.} & \left(K\left(A, C^T\right)u_2 + e_1 b_2\right) + \xi_1 \ge e_1, \\ & \xi_1 \ge 0. \end{array} \tag{2.8}$$

To compute the hyperplanes for nonlinear TWSVM, the dual forms of (2.7) and (2.8) are derived and solved to obtain the hyperplanes in (2.6). However, solving nonlinear TWSVM involves handling two QPPs and requires inverting two matrices of sizes $(m_1 \times m_1)$ and $(m_2 \times m_2)$, which becomes computationally demanding for large datasets. This adds computational complexity compared to the linear case.

2.2. Least squares twin support vector machine (LSTSVM)

The LSTSVM is a binary classification technique that creates two non-parallel hyperplanes, which is proposed by Kumar and Gopal. LSTSVM addresses two modified primal QPPs from TWSVM, applying a least squares approach in which inequality constraints are replaced by equalities in (2.1) and (2.2) as:

$$\begin{aligned} \min_{w_1,b_1,\xi_2} \quad & \frac{1}{2}\|Aw_1 + e_1b_1\|_2^2 + \frac{c_1}{2}\xi_2^T\xi_2 \\ \text{s.t.} \quad & -(Bw_1 + e_2b_1) + \xi_2 = e_2, \end{aligned} \tag{2.9}$$

and

$$\begin{aligned} \min_{w_2,b_2,\xi_1} \quad & \frac{1}{2}\|Bw_2 + e_2b_2\|_2^2 + \frac{c_2}{2}\xi_1^T\xi_1 \\ \text{s.t.} \quad & (Aw_2 + e_1b_2) + \xi_1 = e_1, \end{aligned} \tag{2.10}$$

where c_1, c_2 are the regularization parameters and ξ_1, ξ_2 are slack variables.

In (2.9), the QPP incorporates the L_2-norm of the slack variable ξ_2 with a weight of $\frac{c_1}{2}$ rather than the L_1-norm weighted by c_1 as used in (2.1). This change renders the constraint $\xi_2 \geq 0$ unnecessary. As a result, solving (2.9) reduces to solving a system of linear equations. By substituting the equality constraints directly into the objective function, the problem is rewritten as:

$$\min_{w_1,b_1} \frac{1}{2}\|Aw_1 + e_1b_1\|_2^2 + \frac{c_1}{2}\|Bw_1 + e_2b_1 + e_2\|_2^2. \tag{2.11}$$

Setting the gradient of (2.11) with respect to w_1 and b_1 to zero yields a closed-form solution for QPP (2.9):

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = -\left(F^TF + \frac{1}{c_1}E^TE\right)^{-1} F^Te_2, \tag{2.12}$$

where

$$E = \begin{bmatrix} A & e_1 \end{bmatrix}$$

and

$$F = \begin{bmatrix} B & e_2 \end{bmatrix}.$$

Similarly, solving for the second hyperplane (2.10) gives:

$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = \left(E^TE + \frac{1}{c_2}F^TF\right)^{-1} E^Te_1. \tag{2.13}$$

Therefore, the two nonparallel hyperplanes of LSTSVM can be obtained by inverting two matrices of dimension $(n+1)\times(n+1)$, where n is the number of features, which is significantly smaller than the total number of training samples. This makes LSTSVM more computationally efficient than TWSVM while also improving generalization. The nonlinear case follows the same approach, replacing linear terms with kernel functions.

2.3. Alternating direction method of multipliers (ADMM)

ADMM was first introduced by Glowinski and Marroco [11] and Gabay and Mercier [12]. ADMM is an iterative optimization algorithm designed to decompose complex problems into manageable subproblems, which are then solved alternately. This approach ensures computational efficiency and makes ADMM particularly suitable for distributed and large-scale problems. ADMM solves optimization problems of the form:

$$\begin{aligned} \min_{x,z} \quad & f(x) + g(z) \\ \text{s.t.} \quad & Ax + Bz = c, \end{aligned} \tag{2.14}$$

where f and g are convex functions, and A, B, and c are given matrices/vectors. Since (2.14) is a constrained minimization problem, we can write the related augmented Lagrangian:

$$L_\rho(x, z, y) = f(x) + g(z) + y^T (Ax + Bz - c) + \frac{\rho}{2} \|Ax + Bz - c\|_2^2 .$$

ADMM operates through a sequence of iterations

$$x^{(k+1)} := \operatorname*{argmin}_x L_\rho(x, z^{(k)}, y^{(k)}), \tag{2.15}$$

$$z^{(k+1)} := \operatorname*{argmin}_z L_\rho(x^{(k+1)}, z, y^{(k)}), \tag{2.16}$$

$$y^{(k+1)} := y^{(k)} + \rho(Ax^{(k+1)} + Bz^{(k+1)} - c), \tag{2.17}$$

where $\rho > 0$ is the Lagrangian dual variable, which is also called the penalty parameter. The algorithm involves three key steps. First, an x-minimization step optimizes x using the augmented Lagrangian function L_ρ while keeping z and the dual variable y fixed. Next, a z-minimization step updates z similarly. Finally, the dual variable y is updated using a step size proportional to the augmented Lagrangian parameter ρ.

The Lasso. Lasso regularization, which utilizes the L_1-norm, is an optimization and machine learning approach designed to reduce overfitting and promote sparsity in model parameters and is often solved using the ADMM because ADMM is well-suited for convex optimization problems with separable objective functions and constraints. The corresponding Lasso formulation is expressed as:

$$\min_\beta \quad \frac{1}{2} \|X\beta - y\|_2^2 + \tau \|\beta\|_1 , \tag{2.18}$$

where $y \in \mathbb{R}^n$, $X \in \mathbb{R}^{n \times p}$, and $\tau > 0$ is a scalar regularization parameter that controls the strength of the penalty.

ADMM introduces an auxiliary variable z to separate the least squares and L_1 penalty terms:

$$\begin{aligned} \min_{\beta,z} \quad & \frac{1}{2} \|X\beta - y\|_2^2 + \tau \|z\|_1 \\ \text{s.t.} \quad & \beta - z = 0. \end{aligned} \tag{2.19}$$

ADMM solves this using the augmented Lagrangian:

$$L_\rho(\beta, z, u) = \frac{1}{2} \|X\beta - y\|_2^2 + \tau \|z\|_1 + \frac{\rho}{2} \|\beta - z + u\|_2^2 ,$$

where u is the scaled dual variable and $\rho > 0$ is the penalty parameter controlling convergence. ADMM then alternates between updating β, z, and u:

(1) Update β (least squares step):

$$\beta^{(k+1)} = \left(X^T X + \rho I\right)^{-1} \left(X^T y + \rho\left(z^{(k)} - u^{(k)}\right)\right), \tag{2.20}$$

where $\left(X^T X + \rho I\right)$ is always invertible, since $\rho > 0$.

(2) Update z (soft-thresholding step):

$$z^{(k+1)} = S_{\tau/\rho}\left(\beta^{(k+1)} + u^{(k)}\right), \tag{2.21}$$

where $S_{\tau/\rho}$ is the soft-thresholding operator, which recall is defined as:

$$S_{\varepsilon}(\nu) = \begin{cases} \nu - \varepsilon, & \text{if } \nu > \varepsilon, \\ 0, & \text{if } -\varepsilon \leq \nu \leq \varepsilon, \\ \nu + \varepsilon, & \text{if } \nu < -\varepsilon. \end{cases}$$

(3) Update u (dual variable update):

$$u^{(k+1)} = u^{(k)} + \rho(\beta^{(k+1)} - z^{(k+1)}). \tag{2.22}$$

3. Research methodology

Lasso LSTSVM by ADMM. Building on the foundational concepts discussed earlier, this section presents the formulation for solving LSTSVM with L_1-norm regularization, also known as the Lasso technique, using the ADMM framework. ADMM is chosen over traditional optimization techniques due to its strong scalability and its ability to decompose complex objectives—particularly those involving non-smooth terms like the L_1-norm—into simpler subproblems. This makes it especially effective for high-dimensional, large-scale datasets.

Linear case. In this work, we replace the L_2-norm of the penalty term in LSTSVM (2.9) and (2.10) with the L_1-norm, allowing the problem to be reformulated in a manner similar to the Lasso technique. This modification promotes sparsity in the slack variables and enables the use of ADMM to decompose the problem into simpler subproblems, improving computational efficiency. The modified optimization problems are formulated as follows:

$$\begin{aligned} \min_{w_1, b_1, \xi_2} \quad & \frac{1}{2}\|Aw_1 + e_1 b_1\|_2^2 + \frac{c_1}{2}\|\xi_2\|_1 \\ \text{s.t.} \quad & -(Bw_1 + e_2 b_1) + \xi_2 = e_2, \end{aligned} \tag{3.1}$$

and

$$\begin{aligned} \min_{w_2, b_2, \xi_1} \quad & \frac{1}{2}\|Bw_2 + e_2 b_2\|_2^2 + \frac{c_2}{2}\|\xi_1\|_1 \\ \text{s.t.} \quad & (Aw_2 + e_1 b_2) + \xi_1 = e_1, \end{aligned} \tag{3.2}$$

where c_1, c_2 are given positive parameters.

To facilitate efficient computation, we reformulate the problems using auxiliary variables:

$$\begin{aligned} \min_{x_1, \xi_2} \quad & \frac{1}{2}\|Fx_1\|_2^2 + \tau_1\|\xi_2\|_1 \\ \text{s.t.} \quad & -Ex_1 + \xi_2 = e_2, \end{aligned} \tag{3.3}$$

and

$$\begin{aligned} \min_{x_2,\xi_1} \quad & \frac{1}{2}\|Ex_2\|_2^2 + \tau_2\|\xi_1\|_1 \\ \text{s.t} \quad & Fx_2 + \xi_1 = e_1, \end{aligned} \tag{3.4}$$

where

$$F = \begin{bmatrix} A & e_1 \end{bmatrix}, \quad E = \begin{bmatrix} B & e_2 \end{bmatrix}, \quad x_1 = \begin{bmatrix} w_1 & b_1 \end{bmatrix}^T,$$

and

$$x_2 = \begin{bmatrix} w_2 & b_2 \end{bmatrix}^T.$$

The augmented Lagrangian for the first reformulated problem (e.g., for x_1 and ξ_2) is given by:

$$\begin{aligned} L_\rho(x_1, \xi_2, y_1) = \quad & \frac{1}{2}\|Fx_1\|_2^2 + \tau_1\|\xi_2\|_1 + y_1^T(Ex_1 - \xi_2 + e_2) \\ & + \frac{\rho}{2}\|Ex_1 - \xi_2 + e_2\|_2^2, \end{aligned} \tag{3.5}$$

where $\rho > 0$ serves as a penalty parameter and y_1 is the dual variable. The ADMM update rules for solving this problem are as follows:

(1) x_1**-update:** Solve for $x_1^{(k+1)}$:

$$x_1^{(k+1)} = \underset{x_1}{\operatorname{argmin}}\, L_\rho(x_1, \xi_2^{(k)}, y_1^{(k)}),$$

resulting in:

$$x_1^{(k+1)} = \left(F^TF + \rho E^TE\right)^{-1}\left[E^T\left(\rho\xi_2^{(k)} - \rho e_2 - y_1^{(k)}\right)\right].$$

(2) ξ_2**-update:** Solve for $\xi_2^{(k+1)}$:

$$\xi_2^{(k+1)} = \underset{\xi_2}{\operatorname{argmin}}\, L_\rho(x_1^{(k+1)}, \xi_2, y_1^{(k)}),$$

which simplifies to the soft-thresholding operation:

$$\xi_2^{(k+1)} = S_{\tau_1/\rho}\left(Ex_1^{(k+1)} + e_2 + \frac{y_1^{(k)}}{\rho}\right),$$

where $S_{\tau_1/\rho}(\cdot)$ is the soft-thresholding operator.

(3) **Dual variable update:** Update the dual variable y_1:

$$y_1^{(k+1)} = y_1^{(k)} + \rho\left(Ex_1^{(k+1)} - \xi_2^{(k+1)} + e_2\right).$$

The iterative steps for solving the problem using ADMM are summarized in Algorithm 1:

Algorithm 1 ADMM for solving Lasso LSTSVM 1st plane.

% initialize ξ_2, y_1

$\xi_2^{(0)} \leftarrow \bar{\xi_2}$

$y_1^{(0)} \leftarrow \bar{y_1}$

for $k = 0, 1, 2, \ldots$ **do**

$$x_1^{(k+1)} = \left(F^T F + \rho E^T E\right)^{-1} \left[E^T \left(\rho \xi_2^{(k)} - \rho e_2 - y_1^{(k)}\right)\right]$$

$$\xi_2^{(k+1)} = S_{\tau_1/\rho}\left(E x_1^{(k+1)} + e_2 + \frac{y_1^{(k)}}{\rho}\right)$$

$$y_1^{(k+1)} = y_1^{(k)} + \rho\left(E x_1^{(k+1)} - \xi_2^{(k+1)} + e_2\right)$$

end for

Similarly, the second reformulated problem can be solved using the same concepts as the first problem, as previously demonstrated. The iterative steps for updating x_2, ξ_1, and y_2 are summarized in Algorithm 2:

Algorithm 2 ADMM for solving Lasso LSTSVM 2nd plane.

% initialize ξ_1, y_2

$\xi_1^{(0)} \leftarrow \bar{\xi_1}$

$y_2^{(0)} \leftarrow \bar{y_2}$

for $k = 0, 1, 2, \ldots$ **do**

$$x_2^{(k+1)} = \left(E^T E + \rho F^T F\right)^{-1} \left[-F^T \left(\rho \xi_1^{(k)} - \rho e_1 + y_2^{(k)}\right)\right]$$

$$\xi_1^{(k+1)} = S_{\tau_2/\rho}\left(-F x_2^{(k+1)} + e_1 - \frac{y_2^{(k)}}{\rho}\right)$$

$$y_2^{(k+1)} = y_2^{(k)} + \rho\left(F x_2^{(k+1)} + \xi_1^{(k+1)} - e_1\right)$$

end for

Nonlinear case. In real-world scenarios, the linear kernel method is not always applicable, as large-scale datasets often exhibit higher complexity. Therefore, we extend the algorithm to nonlinear Lasso LSTSVM using kernel techniques [14]. We modify the optimization problems (3.1) and (3.2) as follows:

$$\begin{aligned} \min_{w_1, b_1, \xi_2} \quad & \frac{1}{2}\|K(A, X)\tilde{w}_1 + e_1 b_1\|_2^2 + \frac{c_1}{2}\|\xi_2\|_1 \\ \text{s.t.} \quad & -(K(B, X)\tilde{w}_1 + e_2 b_1) + \xi_2 = e_2, \end{aligned} \tag{3.6}$$

and

$$\begin{aligned} \min_{w_2, b_2, \xi_1} \quad & \frac{1}{2}\|K(B, X)\tilde{w}_2 + e_2 b_2\|_2^2 + \frac{c_2}{2}\|\xi_1\|_1 \\ \text{s.t.} \quad & (K(A, X)\tilde{w}_2 + e_1 b_2) + \xi_1 = e_1. \end{aligned} \tag{3.7}$$

Where

$$K(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^T \phi(\mathbf{y})$$

is a kernel for any $\mathbf{x}$ and $\mathbf{y}$, where function ϕ maps data points $\mathbf{x}$ from $\mathbb{R}^n$ to $\mathbb{R}^m (n < m)$.

The model can be reformulated as the following:

$$\begin{aligned} \min_{x_1,\xi_2} \quad & \frac{1}{2} \|Gx_1\|_2^2 + \tau_1 \|\xi_2\|_1 \\ \text{s.t.} \quad & -Hx_1 + \xi_2 = e_2, \end{aligned} \tag{3.8}$$

and

$$\begin{aligned} \min_{x_2,\xi_1} \quad & \frac{1}{2} \|Hx_2\|_2^2 + \tau_2 \|\xi_1\|_1 \\ \text{s.t} \quad & Gx_2 + \xi_1 = e_1, \end{aligned} \tag{3.9}$$

where

$$G = \begin{bmatrix} K(A, X) & e_1 \end{bmatrix}, \quad H = \begin{bmatrix} K(B, X) & e_2 \end{bmatrix}, \quad x_1 = \begin{bmatrix} \tilde{w}_1 & b_1 \end{bmatrix}^T,$$

and

$$x_2 = \begin{bmatrix} \tilde{w}_2 & b_2 \end{bmatrix}^T.$$

The solution is similar to the linear kernel case: construct the augmented Lagrange function and follow the ADMM framework to update x, ξ and dual variables y.

As for complexity, solving the linear system contributes $O(n^3)$. Our proximal operator is simple. Its complexity is $O(n)$. If k iterations are required, the total complexity is $O(k \cdot (n^3 + n))$. For convex problems, k scales as $O(1/\epsilon)$, so the total complexity becomes $O\left(\frac{n^3}{\epsilon}\right)$. The criteria that ensures that our ADMM converges are the feasibility of the primal and dual. It is important to have feasibility in both primal and the dual (Lagrangian variables).

Accelerated Lasso LSTSVM with guard condition. In this work, we employ an accelerated ADMM framework [15] designed to ensure a consistent reduction in the combined residual γ by introducing a *guard condition*. This condition regulates the acceleration process, allowing it to proceed when met and reverting to the standard ADMM iteration otherwise. This adaptive mechanism prevents potential instability or divergence that might arise from direct acceleration.

Let u denote the target vector we aim to compute. The approximation of u at the $k-$th iteration is represented by $u^{(k)}$. For clarity, $\bar{u}^{(k)}$ represents the approximation of u computed using the standard ADMM method, while $\hat{u}^{(k)}$ denotes the vector derived from an acceleration strategy applied to the ADMM process. In an acceleration step, the next iteration $\hat{u}^{(k+1)}$ is calculated by combining current and previous approximations of $\bar{u}$, often as follows:

$$\hat{u}^{(k)} = \bar{u}^{(k)} + \alpha^{(k)}(\bar{u}^{(k)} - \bar{u}^{(k-1)}), \tag{3.10}$$

where $\alpha^{(k)}$ is an adaptive parameter governing the acceleration. The proposed accelerated ADMM method, incorporating the guard condition, is formally outlined in Algorithm 3.

Algorithm 3 Accelerated Lasso LSTSVM with guard condition.

% initialize ξ_2, y_1

$\xi_2^{(0)} \leftarrow \tilde{\xi}_2$

$y_1^{(0)} \leftarrow \tilde{y_1}$

for $k = 0, 1, 2, \ldots$ **do**

$$x_1^{(k+1)} = \left(F^T F + \rho E^T E\right)^{-1} \left[E^T \left(\rho \xi_2^{(k)} - \rho e_2 - y_1^{(k)}\right)\right];$$

$$\bar{\xi}_2^{(k+1)} = S_{\tau_1/\rho}\left(E x_1^{(k+1)} + e_2 + \frac{y_1^{(k)}}{\rho}\right)$$

$$\bar{y}_1^{(k+1)} = y_1^{(k)} + \rho\left(E x_1^{(k+1)} - \bar{\xi}_2^{(k+1)} + e_2\right)$$

% Begin of acceleration steps

$$\hat{\xi}_2^{(k+1)} = \bar{\xi}_2^{(k+1)} + \alpha_1^{(k)}\left(\bar{\xi}_2^{(k+1)} - \bar{\xi}_2^{(k)}\right)$$

$$\hat{y}_1^{(k+1)} = \bar{y}_1^{(k+1)} + \alpha_1^{(k)}\left(\bar{y}_1^{(k+1)} - \bar{y}_1^{(k)}\right)$$

% End of acceleration steps

$$\gamma^{(k+1)} = \rho^{-1}\left\|\hat{y}_1^{(k+1)} - \hat{y}_1^{(k)}\right\|_2^2 + \rho\left\|\hat{\xi}_2^{(k+1)} - \hat{\xi}_2^{(k)}\right\|_2^2$$

% Begin of guard condition

if $\gamma^{(k+1)} < \gamma^{(0)}\eta^{k+1}$ **then**

$$\xi_2^{(k+1)} = \hat{\xi}_2^{(k+1)}$$

$$y_1^{(k+1)} = \hat{y}_1^{(k+1)}$$

else

$$\xi_2^{(k+1)} = \bar{\xi}_2^{(k+1)}$$

$$y_1^{(k+1)} = \bar{y}_1^{(k+1)}$$

$$\gamma^{(k+1)} = \rho^{-1}\left\|y_1^{(k+1)} - y_1^{(k)}\right\|_2^2 + \rho\left\|\xi_2^{(k+1)} - \xi_2^{(k)}\right\|_2^2$$

end if

% End of guard condition

end for

In our algorithm, the guard condition is designed to monitor the progress of the accelerated ADMM updates and decide whether to accept the acceleration or revert to standard ADMM updates. To make this decision robust and effective, we adopt the parameter selection strategy proposed in [15]. Specifically, the threshold parameter $\eta \in (0, 1)$ is used to determine whether the accelerated update yields sufficient improvement in the combined residual. If the reduction is less than a factor of η, the algorithm rejects the acceleration step and reverts to the previous iterate.

In this work, we set $\eta = 0.85$, following the empirical recommendation in [15]. This choice reflects a balanced trade-off between acceleration and stability. Additionally, the momentum parameter $\alpha^{(k)}$ is used to control the acceleration step size. Following the stationary acceleration approach proposed in [15], we fix

$$\alpha^{(k)} = \alpha$$

for all iterations, rather than using a dynamically updated rule. This simplification reduces computational overhead and avoids potential oscillations caused by increasing momentum.

In particular, [15] provides a convergence proof under the condition that

$$\alpha < 1/3,$$

ensuring the stability of the accelerated scheme. Based on this, we select a fixed value of α within this bound, e.g.,

$$\alpha = 0.2$$

to maintain theoretical convergence guarantees while benefiting from the improved convergence speed that acceleration offers.

This approach achieves faster convergence by leveraging the vectors computed from standard ADMM iterations as a reference for acceleration steps. Notably, selecting appropriate parameters for the guard condition is crucial to optimizing the method's overall efficiency.

4. Results and discussion

This section presents a comparative study evaluating the effectiveness of the accelerated Lasso LSTSVM using ADMM with both linear and nonlinear kernels. We assess the classification accuracy and computational efficiency of the standard LSTSVM, Lasso LSTSVM, and its accelerated variant.

For experiments with linear kernels, we utilize datasets from the UCI machine learning repository [4], including ionosphere, breast cancer, Pima Indians, dry bean, satellite, predict students' dropout and academic success (PSDA), and QSAR biodegradation. For nonlinear classification, we employ EEG eye state and magic telescope (also from the UCI repository), along with the moon dataset—a synthetic dataset from Kaggle [16]—and the electricity dataset [17], a real-world dataset obtained from OpenML.

All datasets are designed for binary or multi-class classification tasks and span diverse domains such as medicine, physics, and social sciences. Their varying characteristics in terms of sample size, feature dimension, and complexity provide a robust foundation for evaluating the generalization performance of the proposed models across different scenarios.

As illustrated in Figure 1, a synthetic dataset was constructed to evaluate model performance on linearly separable data. The dataset consists of 500 samples with 2 features and 2 classes. Orange points represent Class 1, while blue points represent Class 2. To simulate challenging conditions, we introduced outliers by mislabeling 10% of the data—an intentionally high proportion, considering that outliers typically account for less than 5% of real-world datasets.

We then evaluated the models by splitting the dataset into 80% training and 20% testing data. The results show that Lasso LSTSVM achieved an accuracy of 94%, compared to 92% from the standard LSTSVM. This highlights the Lasso model's robustness to label noise and its ability to generalize better in the presence of significant outlier contamination.

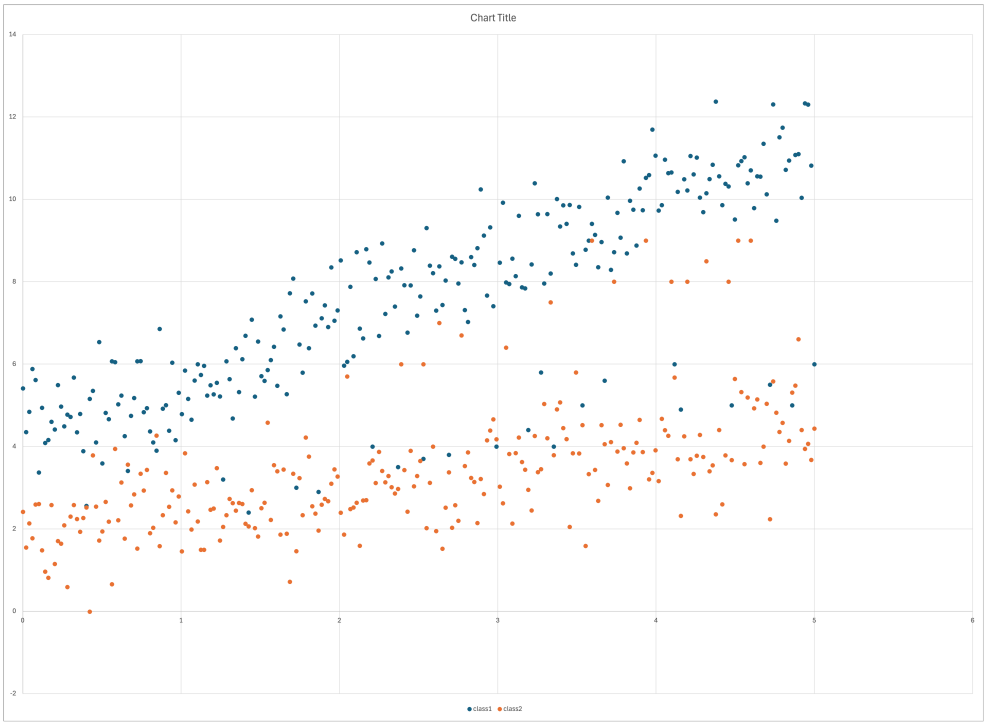

Figure 1. A synthetic data set with outlier which led to misclassification.

The classification results are influenced by the selection of penalty parameters, which are set to specific values for linear and nonlinear kernels to ensure a fair comparison. For nonlinear classifiers, the Gaussian RBF kernel is utilized, with the kernel function defined as

$$K(x_i, x_j) = \exp(-\left\|x_i - x_j\right\| / 2\sigma^2),$$

with the kernel parameter $\sigma = 1$ due to its well-established capability to model complex, nonlinear relationships in high-dimensional feature spaces. It is a widely used choice in kernel-based learning methods because it introduces locality and smoothness in decision boundaries, which is beneficial for datasets with intricate structures, such as EEG eye state and magic telescope.

In our experiments, we fixed the kernel parameter to $\sigma = 1$ for consistency and to avoid additional hyperparameter tuning that could overshadow the performance comparison between models. However, we acknowledge that the performance of RBF-based models is sensitive to the choice of σ. A very small σ may lead to overfitting by capturing noise, while a large σ can oversmooth decision boundaries and underfit the data. The experiment results are summarized in Tables 1–4, where m and n indicate the number of training samples and feature dimensions, respectively.

The datasets used in this study exhibit varying degrees of outlier presence. Based on the interquartile range (IQR) method [18], most datasets were found to contain a moderate to high number of outliers. Notably, the Pima Indians and QSAR datasets have a particularly high concentration of outliers. For Pima Indians, this is primarily due to missing or implausible values treated as real numbers—features such as insulin, BMI, skin thickness, and blood pressure often contain zeros, which are physiologically impossible and act as placeholders for missing data but are not properly handled in the raw dataset.

The QSAR dataset, on the other hand, naturally includes outliers due to its chemical diversity, high dimensionality, and non-normal feature distributions.

Table 1 compares the classification accuracy of the standard LSTSVM and the Lasso LSTSVM using a 10-fold cross-validation approach. Results show that Lasso LSTSVM outperforms the standard LSTSVM on datasets such as ionosphere, Pima Indians, PSDA, and QSAR, all of which contain relatively high levels of outliers. In contrast, for datasets like breast cancer, dry bean, and satellite, which exhibit fewer outliers, the performance gap between the models is narrower.

Table 1. Accuracy ± std (%) comparisons of two algorithms using linear kernel.

Dataset ($m \times n$)	Standard LSTSVM	Lasso LSTSVM
Ionosphere (351×33)	80.00 ± 4.67	90.91 ± 5.70
Breast cancer (558×5)	92.47 ± 2.04	91.58 ± 2.55
Pima indians (768×8)	76.71 ± 6.27	85.53 ± 6.27
Dry Bean (5573×16)	97.78 ± 0.53	97.76 ± 0.68
Satellite (5100×36)	99.27 ± 0.32	99.18 ± 0.34
PSDA (3630×36)	90.77 ± 1.50	91.19 ± 1.24
QSAR (1055×42)	86.13 ± 4.69	87.94 ± 3.05

As shown in Table 2, aside from the moon dataset, which is synthetic and generated from Kaggle, the remaining three datasets (EEG eye state, magic telescope, and electricity) contain a considerable number of outliers. Interestingly, the standard LSTSVM performs better than the Lasso LSTSVM on the Moon dataset, while Lasso LSTSVM shows higher accuracy for EEG eye state and magic telescope. However, there are cases—such as the electricity dataset—where the standard LSTSVM still performs better despite the presence of outliers. This can be attributed to the nature of the outliers in this dataset, which result from natural fluctuations rather than sensor errors or data entry mistakes. In such cases, the L_2-norm of the standard LSTSVM may better capture the underlying data structure, whereas the L_1-norm regularization of Lasso LSTSVM might overly penalize these variations, potentially missing broader trends.

Tables 3 and 4 compare the computational time between the Lasso LSTSVM and the Accelerated Lasso LSTSVM models. The results clearly demonstrate that, for both linear and Gaussian RBF kernels, the accelerated Lasso LSTSVM significantly reduces computation time while maintaining classification accuracy comparable to that of the standard Lasso LSTSVM. This highlights the effectiveness of the proposed acceleration strategy, particularly for large-scale or high-dimensional datasets where computational efficiency is critical.

Table 2. Accuracy ± std (%) comparisons of two algorithms using Gaussian kernel.

Dataset ($m \times n$)	Standard LSTSVM	Lasso LSTSVM
Moon (200×3)	97.50 ± 4.86	95.50 ± 6.43
EEG eye state (14979×14)	84.61 ± 0.77	85.10 ± 0.76
Magic telescope (19020×10)	76.82 ± 0.61	77.08 ± 0.62
Electricity (45312×6)	75.89 ± 0.62	73.08 ± 0.81

Table 3. Performance comparisons of two algorithms using linear kernel.

Dataset ($m \times n$)	Lasso LSTSVM	Accelerated Lasso LSTSVM
	Acc + std(%) time (Sec.)	Acc + std(%) time (Sec.)
Ionosphere (351×33)	90.91 ± 5.70 1.12	85.71 ± 5.33 0.84
Breast cancer (558×5)	91.58 ± 2.55 4.32	91.22 ± 2.46 4.50
Pima indians (768×8)	85.53 ± 6.27 4.93	85.53 ± 6.26 4.42
Dry Bean (5573×16)	97.76 ± 0.68 10.49	97.54 ± 0.54 6.80
Satellite (5100×36)	99.18 ± 0.34 37.49	99.31 ± 0.35 7.25
PSDA (3630×36)	91.19 ± 1.24 9.83	91.74 ± 1.44 3.95
QSAR (1055×42)	87.94 ± 3.05 4.37	86.96 ± 3.93 2.50

Table 4. Performance comparisons of two algorithms using Gaussian kernel.

Dataset ($m \times n$)	Lasso LSTSVM	Accelerated Lasso LSTSVM
	Acc + std(%) time (Sec.)	Acc + std(%) time (Sec.)
Moon (200×3)	95.50 ± 6.43 1.91	96.00 ± 6.58 1.82
EEG eye state (14979×14)	85.10 ± 0.76 141.26	85.69 ± 0.62 117.74
Magic telescope (19020×10)	77.08 ± 0.62 116.39	77.18 ± 0.75 80.15
Electricity (45312×6)	73.08 ± 0.81 65.76	72.86 ± 0.96 84.96

It is widely understood that in regression, ridge regression is used when many predictors or independent variables may be relevant, particularly in multicollinear contexts. In contrast, Lasso regression is employed when we suspect that only a few predictors are significant, allowing for effective feature selection. However, ridge regularization is more sensitive to outliers because it minimizes squared errors, which can lead to biased coefficients that are heavily influenced by those outliers. This principle applies to both the standard LSTSVM models and the Lasso LSTSVM models. Consequently, existing LSTSVM models are likely more sensitive to outliers than the Lasso LSTSVM models.

However, in some instances, the standard LSTSVM models outperformed the Lasso LSTSVM. This may be due to the standard LSTSVM's ability to handle a larger number of relevant predictors or independent variables more effectively. The standard LSTSVM shrinks the coefficients of irrelevant features without eliminating any coefficients entirely. As a result, if the dataset contains many irrelevant features, using the standard LSTSVM may lead to a complex model with numerous included features since none of the irrelevant coefficients are reduced to zero. In contrast, the standard LSTSVM might perform better than the Lasso LSTSVM when the dataset includes relevant independent variables, as shown in some of ours examples.

5. Conclusions

This research presents a comparative analysis between three classification models: the standard LSTSVM, the Lasso LSTSVM, and the proposed accelerated Lasso LSTSVM, using the ADMM framework. The study compares the accuracy of the standard LSTSVM with Lasso LSTSVM and compares the computational time between Lasso LSTSVM and Accelerated Lasso LSTSVM. The inclusion of $L1$-norm regularization helps make the model more robust to outliers, enabling it to adapt well to datasets with a high number of outliers. Additionally, the acceleration step in the ADMM process helps reduce computation time without compromising classification accuracy.

Experimental results on several benchmark datasets, both linear and nonlinear, show that the proposed model outperforms the standard LSTSVM in terms of accuracy. Although there are cases where the standard LSTSVM achieves better accuracy than the Lasso type, this can be attributed to various factors revealed through analysis. However, there remains potential for improvement in computational efficiency, particularly for large-scale datasets. The accelerated model effectively reduces the number of iterations and training time compared to the non-accelerated version. Furthermore, the guard condition applied with the acceleration step ensures the algorithm's stability and guarantees reliable results. This study demonstrates that combining robustness to outliers with ADMM tuning and acceleration steps produces an efficient model that is well-suited for real-world data applications. Future research could extend this work by developing adaptive acceleration techniques and investigating theoretical convergence guarantees to achieve even faster and more reliable algorithms for practical use.

Author contributions

Thanasak Mouktonglang: conceptualization, methodology, validation, formal analysis, investigation, data curation, writing review and editing, supervision, funding acquisition; Rujira

Fongmun: methodology, software, validation, investigation, data curation, writing original draft, writing review and editing. All authors have read and agreed to the published version of the manuscript.

Use of Generative-AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This research was partially supported by the Chiang Mai University, Faculty of Science, Chiang Mai University. The first author was supported by the Development and Promotion of Science and Technology Talents Project (DPST) scholarship.

Conflict of interest

The authors declare no conflicts of interest.

References

1. C. J. Burges, A tutorial on support vector machines for pattern recognition, *Data Min. Knowl. Disc.*, **2** (1998), 121–167. https://doi.org/10.1023/A:1009715923555
2. O. L. Mangasarian, E. W. Wild, Multisurface proximal support vector machine classification via generalized eigenvalues, *IEEE Trans. Pattern Anal. Mach. Intell.*, **28** (2005), 69–74. https://doi.org/10.1109/TPAMI.2006.17
3. Jayadeva, R. Khemchandani, S. Chandra, Twin support vector machines for pattern classification, *IEEE Trans. Pattern Anal. Mach. Intell.*, **29** (2007), 905–910. https://doi.org/10.1109/TPAMI.2007.1068
4. P. M. Murphy, D. W. Aha, *UCI repository of machine learning databases*, University of California, 1992.
5. M. A. Kumar, M. Gopal, Least squares twin support vector machines for pattern classification, *Expert Syst. Appl.*, **36** (2009), 7535–7543. https://doi.org/10.1016/j.eswa.2008.09.066
6. J. A. Suykens, J. Vandewalle, Least squares support vector machine classifiers, *Neural Process. Lett.*, **9** (1999), 293–300. https://doi.org/10.1023/A:1018628609742
7. S. Gao, Q. Ye, N. Ye, 1-norm least squares twin support vector machines, *Neurocomputing*, **74** (2011), 3590–3597. https://doi.org/10.1016/j.neucom.2011.06.015
8. H. Yan, Q. Ye, T. A. Zhang, D. J. Yu, X. Yuan, Y. Xu, et al., Least squares twin bounded support vector machines based on L1-norm distance metric for classification, *Pattern Recognit.*, **74** (2018), 434–447. https://doi.org/10.1016/j.patcog.2017.09.035
9. C. Wang, Q. Ye, P. Luo, N. Ye, L. Fu, Robust capped L1-norm twin support vector machine, *Neural Networks*, **114** (2019), 47–59. https://doi.org/10.1016/j.neunet.2019.01.016

10. L. Yang, Y. Wang, G. Li, Robust capped L1-norm projection twin support vector machine, *J. Ind. Manage. Optim.*, **19** (2023), 5797–5815. https://doi.org/10.3934/jimo.2022195

11. R. Glowinski, A. Marroco, Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problémes de Dirichlet non linéaires, *Anal. Numer.*, **9** (1975), 41–76.

12. D. Gabay, B. Mercier, A dual algorithm for the solution of nonlinear variational problems via finite element approximation, *Comput. Math. Appl.*, **2** (1976), 17–40. https://doi.org/10.1016/0898-1221(76)90003-1

13. C. Chen, R. H. Chan, S. Ma, J. Yang, Inertial proximal ADMM for linearly constrained separable convex optimization, *SIAM J. Imag. Sci.*, **8** (2015), 2239–2267. https://doi.org/10.1137/15100463X

14. K. R. Müller, S. Mika, K. Tsuda, B. Schölkopf, An introduction to kernel-based learning algorithms, *IEEE Trans. Neural Networks*, **12** (2001), 181–201. https://doi.org/10.1109/72.914517

15. A. Buccini, P. Dell'Acqua, M. Donatelli, A general framework for ADMM acceleration, *Numer. Algorithms*, **85** (2020), 829–848. https://doi.org/10.1007/s11075-019-00839-y

16. E. Makhlouf, Linearly inseparable dataset, 2023. Available from: `https://www.kaggle.com/datasets/emadmakhlouf/linearly-inseperable-dataset`.

17. M. Harries, J. Gama, A. Bifet, Electricity dataset, 2009. Available from: `https://www.openml.org/d/151`.

18. F. M. Dekking, *A modern introduction to probability and statistics*, Springer Science & Business Media, 2005. https://doi.org/10.1007/1-84628-168-7

19. O. L. Mangasarian, D. R. Musicant, Lagrangian support vector machines, *J. Mach. Learn. Res.*, **1** (2001), 161–177.

20. S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, IEEE Xplore, 2011. https://doi.org/10.1561/2200000016

21. T. Goldstein, B. O'Donoghue, S. Setzer, R. Baraniuk, Fast alternating direction optimization methods, *SIAM J. Imag. Sci.*, **7** (2014), 1588–1623. https://doi.org/10.1137/120896219

AIMS Press