# LAPLACIAN TWIN SUPPORT VECTOR MACHINE WITH SEMI-SUPERVISED TECHNIQUE FOR PATTERN CLASSIFICATION

VIPAVEE DAMMINSED

A Thesis Submitted to the Graduate School of Naresuan University
in Partial Fulfillment of the Requirements
for the Doctor of Philosophy Degree in Mathematics
March 2024

This thesis entitled "Laplacian twin support vector machine with semi-supervised technique for pattern classification"

by Vipavee Damminsed

has been approved by the Graduate School as partial fulfillment of the requirements for the Doctor of Philosophy Degree in Mathematics of Naresuan University

**Oral Defense Committee**

Suthep Suantai ..... Chair

(Professor Suthep Suantai, Ph.D.)

Rabian Wangkeeree ..... Advisor

(Professor Rabian Wangkeeree, Ph.D.)

Somyot Plubtieng ..... Internal Examiner

(Professor Somyot Plubtieng, Ph.D.)

A-Kaewcharoen ..... Internal Examiner

(Associate Professor Anchalee Kaewcharoen, Ph.D.)

Poom Kum ..... External Examiner

(Professor Poom Kumam, Ph.D.)

**Approved**

Krongkarn Chootip

(Associate Professor Krongkarn Chootip, Ph.D.)

Dean of the Graduate School

13 MAR 2024

# ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to my advisor, Prof. Rabian Wangkeeree, for his support, guidance, and mentorship not only during the research process but also in matters of life advice. I am truly thankful for this opportunity to learn and grow under his good supervision, and especially for always believing in my potential.

My deepest appreciation goes to my family, friends, and colleagues at the laboratory, for being a constant source of inspiration and for the moments that made this academic journey both enriching and enjoyable. Their support has been the driving force behind my achievements.

I extend my gratitude to the Science Achievement Scholarship of Thailand (SAST) for their invaluable financial assistance, which has been instrumental in supporting me throughout my academic journey, from my undergraduate studies to my PhD.

Finally, I would like to take a moment to express my gratitude to myself for working hard. It took concentrate, patience, and a lot of effort to complete this thesis. This achievement reminds me to value my efforts, highlighting the importance of recognizing and appreciating the work that I have dedicated.

Vipavee Damminsed

| | |
|---|---|
| **Title** | LAPLACIAN TWIN SUPPORT VECTOR MACHINE WITH SEMI-SUPERVISED TECHNIQUE FOR PATTERN CLASSIFICATION |
| **Author** | Vipavee Damminsed |
| **Advisor** | Professor Rabian Wangkeeree, Ph.D. |
| **Academic Paper** | Ph.D. Dissertation in Mathematics, Naresuan University, 2023. |
| **Keywords** | Support vector machine, Twin support vector machine, Pinball loss function, Generalized pinball loss function, Semi-supervised classification, Gradient descent algorithm, Laplacian twin support vector machine, Uncertain data classification. |

## ABSTRACT

Semi-supervised learning utilizes labeled data and the geometric information in the unlabeled data to construct a model whereas supervised learning makes use of the only label data. So, semi-supervised learning establishes a more reasonable classifier. In recent years, the Laplacian support vector machine (Lap-SVM) has received a lot of interest in the framework of semi-supervised classification. To develop the performance of Lap-SVM, Laplacian twin support vector machine (Lap-TSVM) has shown exceptional performance as an addition to improve the computational complexity. However, dealing with noise sensitivity and instability for resampling due to its hinge loss function is still a challenge. In the first objective of this research, we provide a novel Laplacian twin support vector machine by combining pinball loss function, termed as Lap-PTSVM. This approach effectively addresses the previously mentioned issues, leading to an improvement in the generalization ability of the classifier.

The second aim of this research is to extend twin support vector machine with generalized pinball loss (GPin-TSVM) into the domain of semi-supervised learning by incorporating graph-based approaches. Leveraging the Laplacian term, we formulate two quadratic programming problems (LapGPin-TSVM). Our proposed method resolves the negative impact of noise and serves as a more accurate

and reasonable classifier. Furthermore, this work is considered a generalized case that encompasses both the pinball and hinge loss. Experimental results on various UCI benchmarks demonstrate the effectiveness and robustness of LapGPin-TSVM.

Furthermore, real-world data often suffers from measurement errors, data staleness, and repeated measurements, introducing uncertainty. To account for data uncertainty, we employ the concept of representing each data feature as a multi-dimensional Gaussian distribution. The proposed Support Vector Machines with an $\epsilon$ insensitive zone pinball loss (UPinSVMs) for uncertain data classification offer noise insensitivity, resampling stability, and increased model sparsity. However, specifying the value of $\epsilon$ remains a challenge. To enhance performance, we introduce the generalized pinball loss ($(\epsilon_1, \epsilon_2)$-Mod-Pin-SVM) into uncertain classification, termed UGPinSVMs. We solve the problems by transforming individuals into unconstrained optimization using an efficient stochastic gradient descent algorithm. More specially, we have introduced verified theorems that are related to our approaches and investigated scatter minimization. Finally, the results from several benchmark datasets show that our model outperforms the existing classifier in terms of accuracy and statistical analysis.

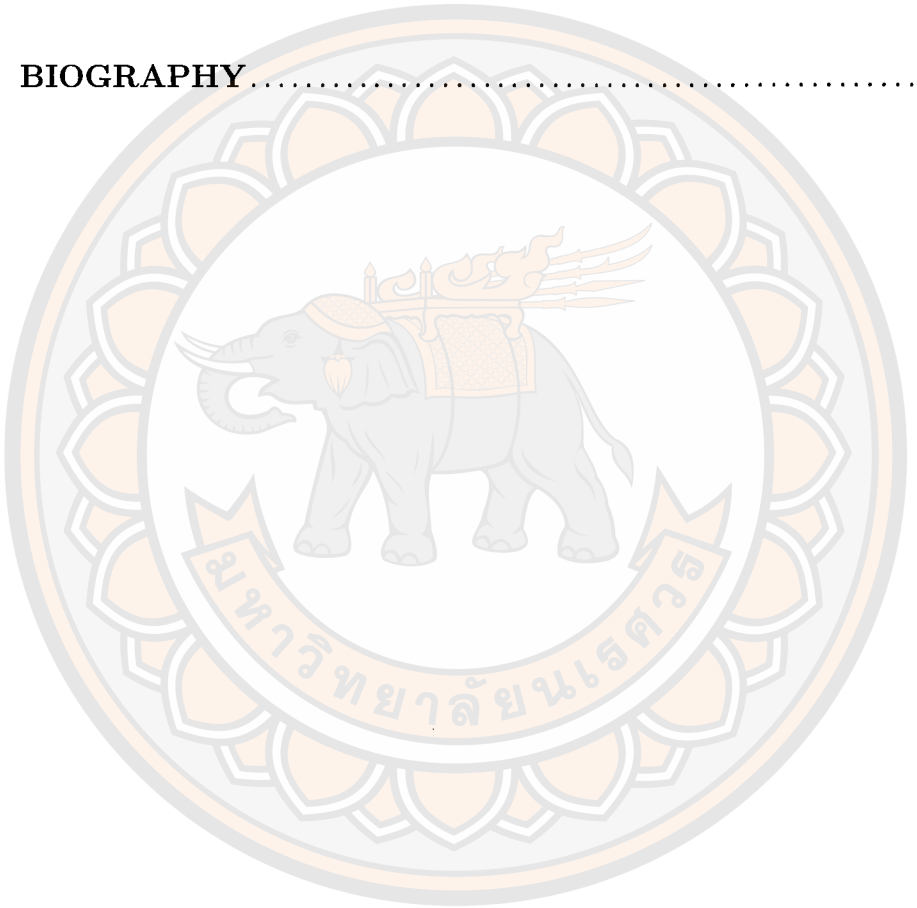
# LIST OF CONTENTS

**Chapter** **Page**

# LIST OF CONTENTS (CONT.)

**Chapter** **Page**





# LIST OF TABLES

**Table** **Page**

# LIST OF TABLES (CONT.)

**Table** **Page**

# LIST OF TABLES (CONT.)

**Table** **Page**

# LIST OF FIGURES

**Figure** **Page**

# LIST OF FIGURES (CONT.)

**Figure** **Page**

# CHAPTER I

# INTRODUCTION

Labeled data plays a crucial role in machine learning, as it is essential for constructing models to predict or forecast information in classification and regression scenarios. A type of machine learning that utilize labeled data is known as supervised learning. Support vector machine (SVM) [1,2] is a popular and commonly use in this framework. SVM is designed based on the statistical learning theory and has a wide range of applications. By seeking optimal decision hyperplanes, SVM maximizes the margin between classes through quadratic programming problem (QPP), enhancing generalization and minimizing errors during training through structural risk minimization (SRM) principles. However, as the amount of data increases, SVM exhibits high computational complexity. To overcome this issue, Jayadeva et al. [3] developed a novel method called twin support vector machine (TSVM) for binary classification. TSVM aims to find two non-parallel hyperplanes, one nearest to each of the two classes, with the other as far away as possible. The problems can be achieved by solving two smaller QPPs separately, making the approach four times faster than SVM, which solves one large QPP.

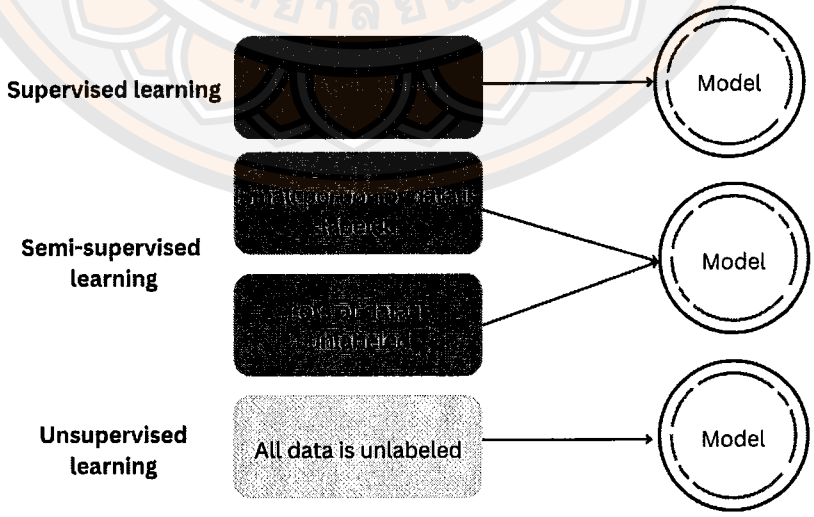



**Figure 1 Illustration of supervised vs unsupervised vs semi-supervised machine learning in a nutshell.**

As mentioned earlier, researchers commonly employ the hinge loss function in support vector machines, which is suffer to noise sensitivity and instability during re-sampling. To address these issues, Huang et al. [4] introduced a novel loss function named pinball loss and applied it to create a method within the SVM framework (Pin-SVM). The core idea of pinball loss is to maximize the quantile distance between samples from the two classes. Subsequently, it was implemented in TSVM, leading to the proposal of several variants [5–7]. Then, Tanveer et al. [8] proposed a twin support vector machine using the pinball loss function (Pin-TSVM) to address the shortcomings of hinge loss-based support vector machines. As a result, the model is less noise-sensitive and exhibits stability re-sampling greater than the TSVM.

However, one main drawback of supervised learning methods is that effective generalization results require a large amount of labeled data. In fact, Labeling is often costly, requiring skilled human agents. A machine learning technique called semi-supervised learning can help enhance classification performance when labeled data is limited and unlabeled data is plentiful [9]. The conclusions drawn from the analysis of the specific type of machine learning are illustrated in Figure 1. By employing in expensive unlabeled input, semi-supervised SVMs [10, 11] have been presented to develop generalization performance. The methods focus on Laplacian spectral technique [12] have gained a lot of attention among the many different forms of semi-supervised SVMs. Laplacian support vector machine (Lap-SVM) [13] have shown outstanding results. Qi et al. [14] introduced Laplacian twin support vector machine (Lap-TSVM), enhancing capabilities for semi-supervised classification and outperforming traditional TSVM in flexibility, prediction accuracy, and generalization.

From the perspective of noise sensitivity and unstable for resampling in the Lap-TSVM , many techniques have been proposed to tackle these problems, such as IFLap-TSVM [15], SIFTSVM [16]. However, there is no theoretical analyst to guarantee the performance of those problems in the current works. To overcome those issues in considering semi-supervised classification learning, we present a model called Laplacian twin support vector machine with pinball loss (Lap-PTSVM) which constructs nonparallel-hyperplanes and utilize the concept of

quantile distance of pinball loss to measure an error.

Recently, Rastogi et al. [17] introduced a novel loss function called the $(\epsilon_1, \epsilon_2)$-insensitive zone pinball loss, serving as a generalization of other existing loss functions, including hinge loss and pinball loss. Subsequently, Panup et al. [18] extended the work of Rastogi et al. to GPin-TSVM, applying the approach to the problem of twin support vector machines. The result effectively addresses challenges such as noise insensitivity and instability in re-sampling. Consequently, in the second study, we aim to extend the approach from GPin-TSVM, originally a supervised model, to a semi-supervised framework by incorporating the Laplacian technique, termed as LapGPin-TSVM. The goal is to enable the model to handle data in situations where labeled data is limited, thereby enhancing the overall performance of the model.

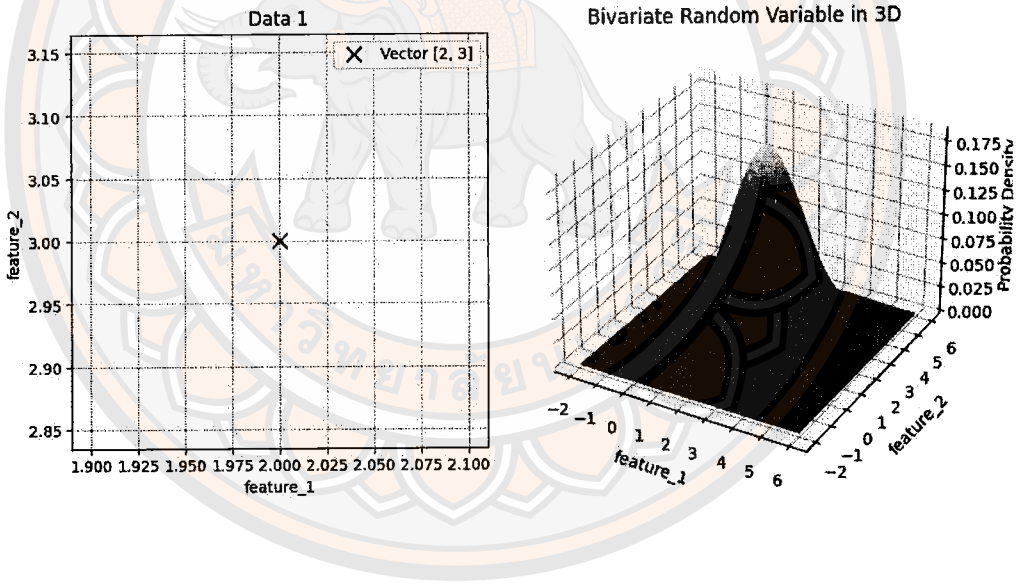



**Figure 2 The two-dimensional data represents the exact values of the features (on the left). Data representation with a bivariate normal distribution, treating data as a random variable rather than using exact feature values (on the right).**

Typically, the models are constructed by learning from the exact values of the data input, assuming that each feature of the data samples can be described by a single value as you can see in Figure 2. However, in real-world problems, measurement and statistical errors during data collection can introduce uncertainty to the data. As a result, alternative learning methods have been proposed to handle

input uncertain data modifications, as described in [19–21].

In SVM studies, Tzelepis et al. [22] introduced SVMs with Gaussian distributions (SVMsG) incorporating hinge loss for measurement errors. However, this approach struggles when uncertain samples lie on class boundaries. Liang et al. [23] proposed innovative SVMs prioritizing noise insensitivity and stability during resampling, utilizing an $\epsilon$-insensitive pinball loss function. This method captures more information from uncertain data points compared to hinge loss, addressing the limitations of SVMsG.

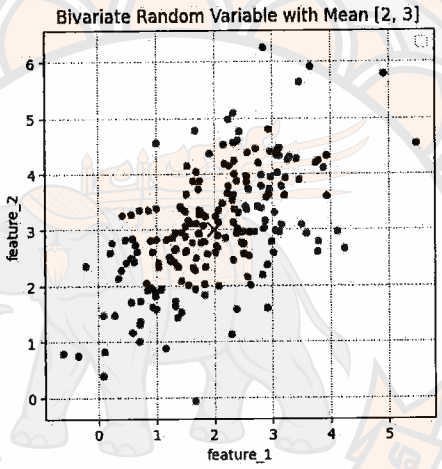



**Figure 3 Illustration of 100 random vectors (data) generate from the bivariate normal distribution with mean vecor [2, 3].**

Motivated by the works of Rastogi [17], Tzelepis [22], and Liang [23], we have been inspired to develop flexible models capable of handling uncertain data. Therefore, in the latest work, we introduce a novel method called UGPinSVMs, which utilizes the concept of generalized pinball loss and employs the multivariate normal distribution to describe the uncertainty of the data. As you can see in Figure 3, it shows the scatter plot of data generated from 100 random vectors sampled from the distribution. This approach exhibits noise insensitivity, is stable for resampling and also generalizes the capacity of model for uncertain data classification

The thesis is organized in the following manner:

**Chapter I.** We discuss the background and significance of the work.

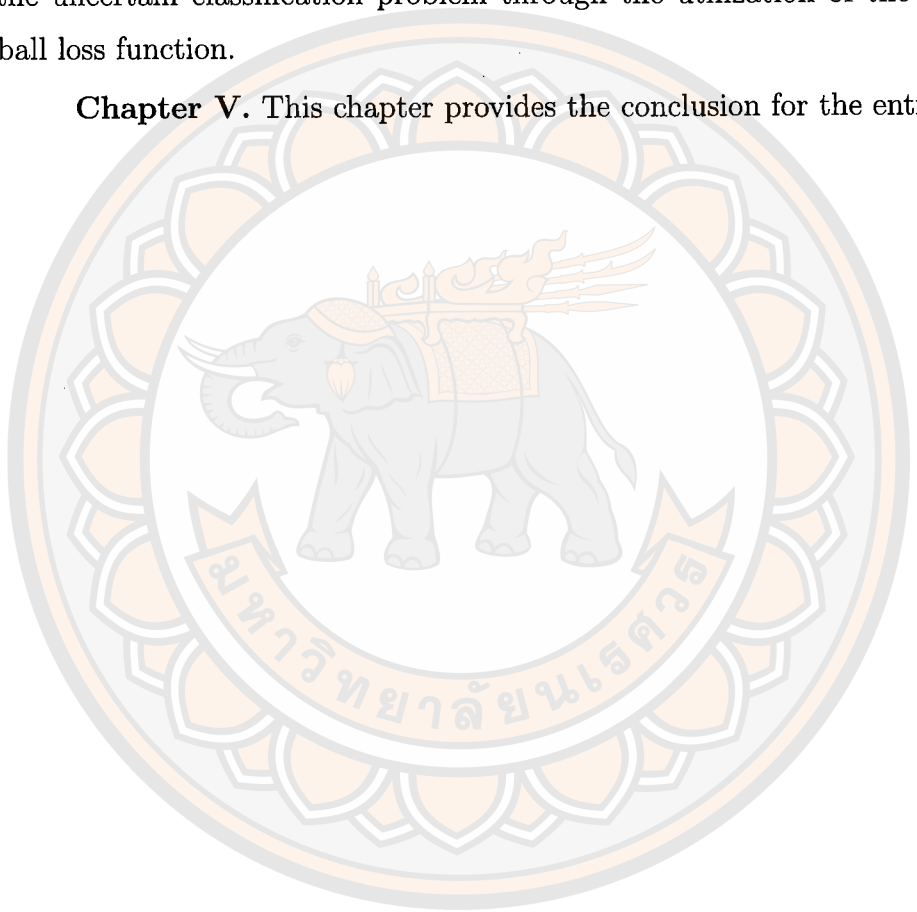
**Chapter II.** In this chapter, we cover notations, fundamental knowledge,

and preliminary results serving as the foundation for this thesis.

**Chapter III.** In this chapter, we introduce Lap-PTSVM, a model that constructs nonparallel planes as an improvement over Lap-TSVM. Additionally, we extend GPin-TSVM, originally a supervised learning method, into the framework of semi-supervised learning, named as LapGPin-TSVM.

**Chapter IV.** In this chapter, our focus is on applying the SVM concept to the uncertain classification problem through the utilization of the generalized pinball loss function.

**Chapter V.** This chapter provides the conclusion for the entire work.



# CHAPTER II

# PRELIMINARIES

In this chapter, we introduce the notations, fundamental knowledge, and preliminary results that form the basis for this research. The discussion includes support vector machines, twin support vector machines, semi-supervised techniques, and the classification of uncertain data.

In this thesis, we explore binary classification within the $n$-dimensional Euclidean space $\mathbb{R}^n$. All vectors are treated as column vectors, with the option to transpose them into row vectors denoted by the superscript $\top$. For vectors $\mathbf{x} = [x_1, x_2, \ldots, x_n]^\top$ and $\mathbf{y} = [y_1, y_2, \ldots, y_n]^\top$ in $\mathbb{R}^n$, the scalar product of $\mathbf{x}$ and $\mathbf{y}$ is expressed as $\mathbf{x}^\top\mathbf{y} = \sum_{i=1}^n x_i y_i$, and the induced norm (Euclidean norm) of $\mathbf{x}$ is defined as $\|\mathbf{x}\| = \mathbf{x}^\top\mathbf{x} = \sqrt{\sum_{i=1}^n x_i^2}$.

## 2.1 Definitions, theorems and notation

**Definition 2.1.1.** [25] Let $A$ be an $n \times n$ real square matrix. A nonzero vector $\mathbf{v}$ is called an **eigenvector** of A if there exists a scalar $\lambda$ such that $A\mathbf{v} = \lambda\mathbf{v}$. The scalar $\lambda$ is called an **eigenvalue** corresponding to eigenvector $\mathbf{v}$.

**Definition 2.1.2.** [25] A **symmetric matrix** is a matrix $A$ such that $A^\top = A$.

**Theorem 2.1.3.** [25] All eigenvalues of a symmetric matrix are real.

**Definition 2.1.4.** [25] A symmetric matrix $A \in \mathbb{R}^n$ is called **positive semidefinite**, denoted by $A \succeq \mathbf{0}$, if $\mathbf{x}^\top A\mathbf{x} \geq 0$ for every $\mathbf{x} \in \mathbb{R}^n$.

**Definition 2.1.5.** [25] A symmetric matrix $A \in \mathbb{R}^n$ is called **positive definite**, denoted by $A \succ \mathbf{0}$, if $\mathbf{x}^\top A\mathbf{x} > 0$ for every $\mathbf{0} \neq \mathbf{x} \in \mathbb{R}^n$.

**Theorem 2.1.6.** [25] A symmetric matrix $A$ is positive definite (or positive semidefinite) if and only if all eigenvalues of $A$ are positive (or nonnegative).

Next, we explain the concepts related to hyperplane and halfspace, which are ideas used in building models for classification.

**Definition 2.1.7.** [26] Let $\mathbf{u}$ be a nonzero vector in $\mathbb{R}^n$ and let $a \in \mathbb{R}$. The set of all point $\mathbf{x} = [x_1, x_2, \ldots, x_n]^\top$ satisfy the linear equation $u_1x_1+u_2x_2+\cdots+u_nx_n = a$ is called a **hyperplane** of the space $\mathbb{R}^n$.

We may describe the hyperplane by $\{\mathbf{x} \in \mathbb{R}^n : \mathbf{u}^\top\mathbf{x} = a\}$, where $\mathbf{u} = [u_1, u_2, \ldots, u_n]^\top$.

**Definition 2.1.8.** [26] Let $\mathbf{u}$ be a nonzero vector in $\mathbb{R}^n$ and let $a \in \mathbb{R}$. The set of all point $\mathbf{x} = [x_1, x_2, \ldots, x_n]^\top$ satisfy the linear equation $u_1x_1+u_2x_2+\cdots+u_nx_n \leq a$ is called a **halfspace** of the space $\mathbb{R}^n$.

We describe a real-valued function that involves the differentiability and convexity of the function. We begin by referring to the partial derivative.

**Definition 2.1.9.** [26] Let $f : \mathbb{R}^n \to \mathbb{R}$ be some function, fix some $\mathbf{x} \in \mathbb{R}^n$. If the limit

$$\lim_{t\to 0^+} \frac{f(\mathbf{x} + t\mathbf{e}_i) - f(\mathbf{x})}{t}$$

exists, ($\mathbf{e}_i$ is the $i$-th unit vector (all components are 0 except for the $i$-th component which is 1)), then it is called the $i$-th partial derivative of $f$ at the vector $\mathbf{x}$ and and is denoted by $\frac{\partial f(\mathbf{x})}{\partial x_i}$.

**Definition 2.1.10.** [26] $A$ set $S \subset \mathbb{R}^n$ is called a **convex set** if for any two vectors $\mathbf{x}_1, \mathbf{x}_2 \in S$ and any $\lambda \in [0, 1]$, we have

$$\lambda\mathbf{x}_1 + (1 - \lambda)\mathbf{x}_2 \in S.$$

**Definition 2.1.11.** [26] Let $f$ be a function from $\mathbb{R}^n$ to $\mathbb{R}$. The function $f$ is called a **convex function** on $\mathbb{R}^n$ if for every $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n$ and $\lambda \in [0, 1]$,

$$f(\lambda\mathbf{x}_1 + (1 - \lambda)\mathbf{x}_2) \leq \lambda f(\mathbf{x}_1) + (1 - \lambda)f(\mathbf{x}_2).$$

The function $f$ is called a **strictly convex function** on $\mathbb{R}^n$ if for any $\mathbf{x}_1 \neq \mathbf{x}_2 \in \mathbb{R}^n$ and $\lambda \in [0, 1]$,

$$f(\lambda\mathbf{x}_1 + (1 - \lambda)\mathbf{x}_2) < \lambda f(\mathbf{x}_1) + (1 - \lambda)f(\mathbf{x}_2).$$

**Definition 2.1.12.** [26] Let $F : C \to \mathbb{R}$ be a function defined over a convex set $C \subseteq \mathbb{R}^n$. The function $F$ is called **strongly convex** over $C$ if there exists $\rho > 0$ such that the function $F(\mathbf{x}) - \frac{\rho}{2}\|\mathbf{x}\|^2$ is convex over $C$.

**Lemma 2.1.13.** [26] Let $f : \mathbb{R}^n \to \mathbb{R}$ be strongly convex. Then the (unique) minimizer of $f$ is a strong minimizer.

**Definition 2.1.14.** [26] Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function and $\bar{\mathbf{x}} \in \mathbb{R}^n$. Then $\boldsymbol{\xi} \in \mathbb{R}^n$ is said to be the **subgradient** of the function $f$ at $\bar{\mathbf{x}}$ if

$$f(\mathbf{x}) \geq f(\bar{\mathbf{x}}) + \langle \mathbf{x} - \bar{\mathbf{x}}, \boldsymbol{\xi} \rangle, \quad \forall \mathbf{x} \in \mathbb{R}^n.$$

The set of subgradients of $f$ at $\bar{\mathbf{x}}$ is called the **subdifferential** of $f$ at $\bar{\mathbf{x}}$ and denoted $\partial f(\bar{\mathbf{x}})$.

**Definition 2.1.15.** [26] Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function. The function denoted by $\nabla f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ is said to be the **gradient** of the function $f$ if

$$\nabla f(\mathbf{x}) = \left[ \frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_n} \right]^\top.$$

**Theorem 2.1.16.** [26] Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable convex function at $\bar{\mathbf{x}}$. Then $\partial f(\bar{\mathbf{x}}) = \{\nabla f(\bar{\mathbf{x}})\}$, where $\nabla f(\mathbf{x}) = \left[ \frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_n} \right]^\top$ is the gradient of the function $f$.

**Corollary 2.1.17.** [26] Consider the quadratic function in $\mathbb{R}^n$

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top H \mathbf{x} + \mathbf{b}^\top \mathbf{x} + c,$$

where $H \in \mathbb{R}^{n \times n}, \mathbf{b} \in \mathbb{R}^n$ and $c \in \mathbb{R}$. If $H$ is positive semidefinite, then $f(\mathbf{x})$ is a convex function in $\mathbb{R}^n$. Similarly, if $H$ is positive definite, then $f(\mathbf{x})$ is a strictly convex function in $\mathbb{R}^n$.

We already define a general constrained optimization problem as well as the properties that are exceptional to convex constrained optimization problems.

**Definition 2.1.18.** [26] Let $f, g_i : X \to \mathbb{R}$, $i = 1, \ldots, m$ be convex functions defined over a set $X \subseteq \mathbb{R}^n$. Then the following problem is a convex constrained optimization problem:

$$\begin{aligned} \min \quad & f(\mathbf{x}), \\ \text{subject to} \quad & g_i(\mathbf{x}) \leq 0, \quad \forall i = 1, \ldots, m. \end{aligned} \tag{2.1.1}$$

**Theorem 2.1.19.** [26] Consider the problem (2.1.1), where the objective function $f(\mathbf{x})$ is strictly convex. Then its solution is unique when it has solution.

**Theorem 2.1.20.** [26] Consider the quadratic programming problem (QPP)

$$\begin{aligned} \min \quad & \frac{1}{2}\mathbf{x}^\top H\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c, \\ \text{subject to} \quad & Ax - b \leq \mathbf{0}, \end{aligned} \tag{2.1.2}$$

where $H \in \mathbb{R}^{n\times n}, \mathbf{b} \in \mathbb{R}^n$ and $c \in \mathbb{R}$. If $H$ is positive semidefinite, then the above problem is a convex programming, i.e. a **convex quadratic programming problem**.

**Definition 2.1.21.** [26] The Lagrange function associated to the general constrained optimization problem defined in (2.1.1) is the function defined over $X\times\Lambda \subset \mathbb{R}^n \times \mathbb{R}_+$ by:

$$L(\mathbf{x}, \boldsymbol{\alpha}) = f(\mathbf{x}) + \sum_{i=1}^{m} \alpha_i g_i(\mathbf{x}), \quad \forall \mathbf{x} \in X, \boldsymbol{\alpha} \geq 0,$$

where the variables $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)^\top$ are known as the Lagrange multipier or dual variables. The Lagrange multipier associated with the $i$ th constraint is represented by the scalar $\alpha_i$.

**Definition 2.1.22.** [26] The dual (optimization) problem associated to the constrained optimization problem is

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \inf_{\mathbf{x}\in X} L(\mathbf{x}, \boldsymbol{\alpha}), \\ \text{subject to} \quad & \boldsymbol{\alpha} \geq 0. \end{aligned} \tag{2.1.3}$$

**Theorem 2.1.23.** [26] The dual problem (2.1.3) is a convex programming problem.

**Definition 2.1.24.** [26] Assume that $\text{int}(X) \neq \emptyset$. Then Slater's condition is defined as

$$\exists \mathbf{x} \in \text{int}(X) \ \text{ s.t. } \ \text{g}(\mathbf{x}) < 0.$$

We give necessary and sufficient optimality conditions of convex constrained optimization problems in the theorem.

**Theorem 2.1.25.** [26] Assume that $f, g_i : X \to \mathbb{R}$, $i = 1, \ldots, m$ are convex functions and additionally the constraints are satisfied the Slater's condition. Then $\mathbf{x}^*$ is a solution of the constrained program if and only if there exists $\boldsymbol{\alpha} \geq 0$ such that,

$$\mathbf{0} \in \partial f(\mathbf{x}^*) + \sum_{i=1}^{m} \alpha_i \partial g_i(\mathbf{x}^*),$$

$$g_i(\mathbf{x}^*) \leq 0 \quad i = 1, \cdots, m,$$

$$\alpha_i g_i(\mathbf{x}^*) = 0 \quad i = 1, \cdots, m.$$

The above conditions are known as the KKT condition.

**Theorem 2.1.26.** [26] Consider the convex programming problem (2.1.3). If $\mathbf{x}^*$ satisfies the KKT conditions, then $\mathbf{x}^*$ is its solution.

**Theorem 2.1.27.** [26] Consider the unconstrained convex optimization problem:

$$\begin{aligned} \min \quad & f(\mathbf{x}), \\ \text{subject to} \quad & \mathbf{x} \in \mathbb{R}^n, \end{aligned} \tag{2.1.4}$$

then $\bar{\mathbf{x}} \in \mathbb{R}^n$ is an optimal solution of the above problem if and only if $\mathbf{0} \in \partial f(\bar{\mathbf{x}})$

## 2.2 Support vector machine (SVM)

Support Vector Machine (SVM) as originally proposed by Vladimir Vapnik [1,2] within the area of statistical learning theory and structural risk minimization, have demonstrated to work successfully on various classification and forecasting problems. The concept of construting the SVM are illustrate in Figure 4. We call the vector on the support line is support vector.

Suppose we have $m$ training samples like $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)\}$ where $\mathbf{x}_i \in \mathbb{R}^n$ and $y_i \in \{+1, -1\}$. The equation (2.2.1) represents the hyperplane used for data portioning in SVM.

$$\mathbf{w}^\top \mathbf{x} + b = 0, \tag{2.2.1}$$

where $\mathbf{w}$ is a weight, $\mathbf{x}$ is the training sample and $b$ is the bias of the hyperplane. The margin between two classes are to be maximized. The optimization problem can be defined as

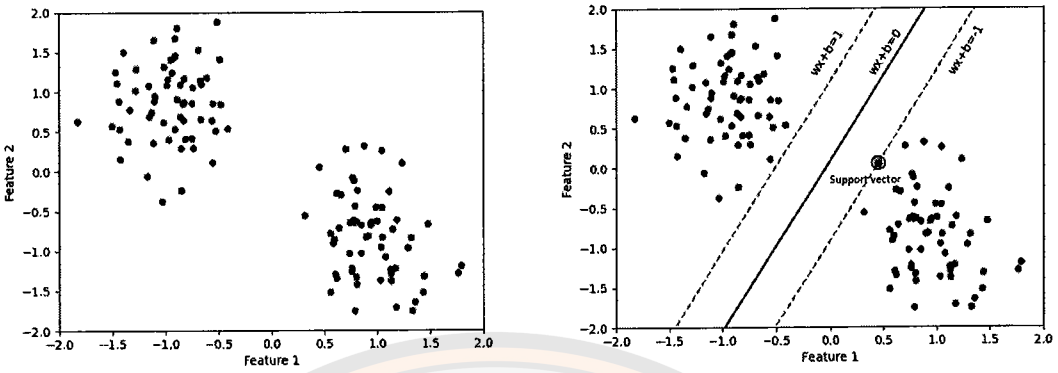



**Figure 4 Geometric interpretation of linearly separable problem and the idea of separating line with maxima margin.**

$$\max_{\mathbf{w},b} \quad \frac{2}{\|\mathbf{w}\|} \qquad (2.2.2)$$
$$\text{s.t.} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad \text{for } i = 1, \ldots, m,$$

or

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|^2 \qquad (2.2.3)$$
$$\text{s.t.} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad \text{for } i = 1, \ldots, m.$$

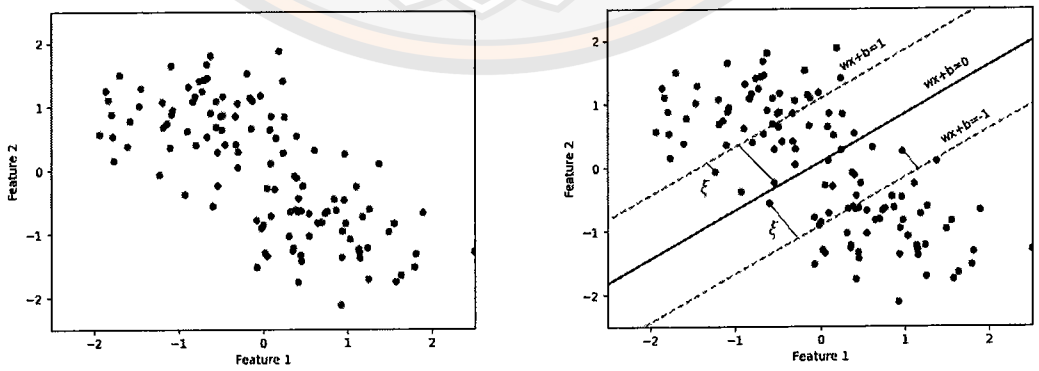



**Figure 5 Geometric interpretation of Non-linearly separable problem and the idea of slack variable.**

For linearly non-separable data as shown in Figure 5, we introduce a non-negative variable called slack variable $\xi_i$ to the objective function resulting in changing the primal problem (2.2.3) into

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i, \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \quad i = 1, \ldots, m, \end{aligned} \tag{2.2.4}$$

where $\boldsymbol{\xi} = (\xi_1, ..., \xi_m)^\top$, and $C > 0$ is penalty parameter. To obtain the solution of (2.2.4), the problem is converted into the dual problem. By introducing the Lagrange multiplier $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_m) \geq \mathbf{0}$ and $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_m) \geq \mathbf{0}$, then we get the Lagrangian function corresponding to the problem as follows:

$$L(\mathbf{w}, b, \boldsymbol{\beta}, \boldsymbol{\alpha}) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\beta_i(y_i(\mathbf{x}_i^\top \mathbf{w} + b) - 1 + \xi_i) - \sum_{i=1}^{m}\alpha_i\xi_i. \tag{2.2.5}$$

Then, we have the following problem:

$$\begin{aligned} \min \quad & \frac{1}{2}\sum_{i,j=1}^{m}\beta_i\beta_j(\mathbf{x}_i^\top \mathbf{x}_j)y_i y_j - \sum_{i=1}^{m}\beta_i \\ \text{subject to} \quad & \sum_{i=1}^{m}\beta_i y_i = 0, \\ & 0 \leq \beta_i \leq C, i = 1, \ldots, m. \end{aligned} \tag{2.2.6}$$

After obtaining the solution to equation (2.2.4), this criterion can be used to predict a new sample $\mathbf{x}$,

$$y = \text{sign}(\mathbf{w}^\top \mathbf{x} + b). \tag{2.2.7}$$

In the framework of machine learning, the concept of kernels plays an essential role in enhancing the expressiveness and flexibility of algorithms, particularly in the context of support vector machine (SVM) and related techniques. A kernel can be viewed as a mathematical function that implicitly transforms data into a higher-dimensional space, allowing algorithms to capture intricate patterns and relationships that may not be discernible in the original feature space.

**Definition 2.2.1.** A function $K : X \times X \to \mathbb{R}$ is called a kernel over $X$, if

$$K(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle,$$

for any $\mathbf{x}, \mathbf{y} \in X$ and for some function $\phi$ from $X$ to a Hilbert space $H$ called a feature space.

The following example gives kernel function commonly used in application.

**Example 2.2.2. (Linear kernel)** The linear kernel is the kernel $K$ defined over $\mathbb{R}^n$ as

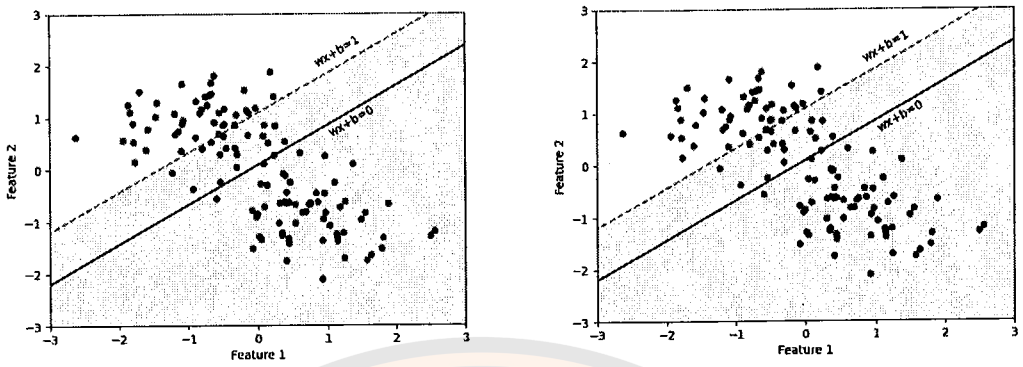
$$K(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y}.$$

**Example 2.2.3. (Gaussian kernel)** For any constant $\sigma > 0$, a Gaussian kernel or radial basis function (RBF) is the kernel $K$ defined over $\mathbb{R}^n$ as

$$K(\mathbf{x}, \mathbf{y}) = \exp\Big( - \frac{\|\mathbf{x} - \mathbf{y}\|^2}{2\sigma^2} \Big).$$

## 2.3 Loss function

It is important to note that SVM and its extensions construct classifiers by uniformly employing and assigning equal significance to misclassified data. The original SVM, which is based on the concept of slack variables, incorporates the hinge loss function. An illustration of computing the hinge loss is depicted in Figure 6. However, this approach can render them sensitive to noise, outliers, and class imbalances [32], which could lead to reduced predictive accuracy or susceptibility to overfitting problems in the model.




**(a)** Hinge loss. **(b)** Pinball loss.

**Figure 6 Geometric interpretation of the concept involves computing the hinge loss and pinball loss.**

The concept of a slack variable in support vector machines is equivalent to the hinge loss function, which is defined by $\mathcal{L}_{hinge}(v) = \max\{0, v\}$ where $v = 1 - y_i(\mathbf{w}^\top \mathbf{x} + b)$. Due to this loss penalty, only the misclassified samples contribute to the problem, causing it to be sensitive to noise.

To handle data corrupted by noise, Huang et al. [4] introduced the pinball loss function to SVM (Pin-SVM). This model penalizes all data which is also correctly identified samples, as demonstrated by the pinball loss function, which is formally defined as:

$$\mathcal{L}_{pin}(v) = \begin{cases} v, & v \geq 0, \\ -\tau v, & v < 0, \end{cases} \tag{2.3.1}$$

where $v = 1 - y_i(\mathbf{w}^\top \mathbf{x} + b)$, $\tau \geq 0$. The concept of computing this loss is shown in Figure 6. Although the above formulation achieves noise insensitivity, it loses sparsity in the process. This is because the sub-gradient of pinball loss function is non-zero almost everywhere.

To address this issue, Huang et al. [4] developed Pin-SVMs to an $\epsilon$-insensitive zone ($\epsilon$-insensitive zone pinball SVMs), resulting in the sub-gradient function turns out to be zero in the range $[\tau, \epsilon]$, providing sparsity back to the

process in model. The $\epsilon$-insensitive zone is defined by:

$$\mathcal{L}_{pin}^{\epsilon}(v) = \begin{cases} v - \epsilon, & v > \epsilon, \\ 0, & -\frac{\epsilon}{\tau} \leq v \leq \epsilon, \\ -\tau\left(v + \frac{\epsilon}{\tau}\right), & v < -\frac{\epsilon}{\tau}, \end{cases} \tag{2.3.2}$$

where $v = 1 - y(\mathbf{w}^\top \mathbf{x} + b)$ and $\tau, \epsilon \geq 0$. The $\epsilon$-insensitive zone follows the pinball loss when substitutes $\epsilon = 0$. Therefore, this loss inherits noise insensitivity and stability for resampling from Pin-SVM. Additionally, it can solve the sparsity loss problem. Utilizing the $\epsilon$-insensitive loss has many beneficial properties.

However, in order to improve the performance and enhance the capabilities of the model, Rastogi et al. [17] introduced the Modified $(\epsilon_1, \epsilon_2)$-insensitive zone support vector machine with pinball loss call generalized pinball loss SVM, which can be considered as an optimal insensitive zone Pin-SVM. This is encouraged from the notion of $\epsilon$-insensitive zone Pin-SVM. Their method provides an asymmetric insensitive zone by allowing a variable $\epsilon$. The definition of the generalized pinball loss function is given as follows

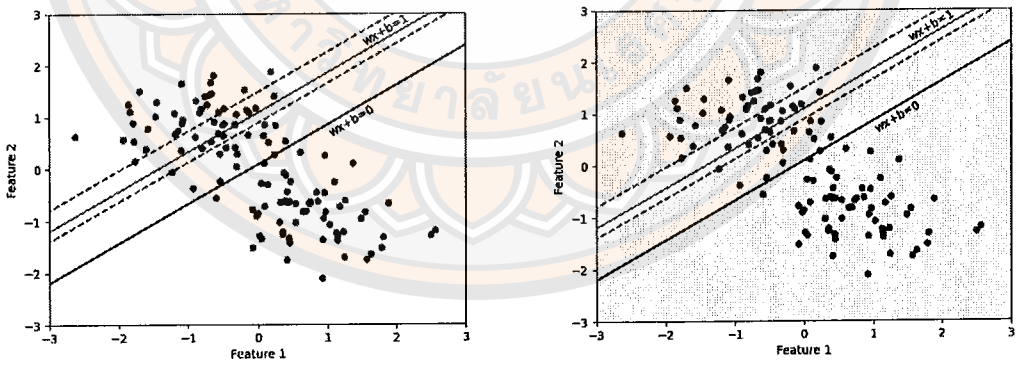



**Figure 7 Geometric interpretation of the concept involves computing the $\epsilon$-insensitive.**

$$\mathcal{L}_{\tau_1,\tau_2}^{\epsilon_1,\epsilon_2}(v) = \begin{cases} \tau_1\big(v - \frac{\epsilon_1}{\tau_1}\big), & v > \frac{\epsilon_1}{\tau_1}, \\ 0, & -\frac{\epsilon_2}{\tau_2} \leq v \leq \frac{\epsilon_1}{\tau_1}, \\ -\tau_2\big(v + \frac{\epsilon_2}{\tau_2}\big), & v < -\frac{\epsilon_2}{\tau_2}, \end{cases} \tag{2.3.3}$$

where $\tau_1, \tau_2, \epsilon_1, \epsilon_2 \geq 0$. The generalized pinball loss follows the $\epsilon$-insensitive zone when substitutes $\epsilon_1 = 0$, $\epsilon_2 = 0$ and $\tau_1 = 1$. It improves the sparsity of the model in a manner similar to the $\epsilon$-insensitive zone, still has all of the properties based on the $\epsilon$-insensitive zone, and also generalized the ability of the model.

## 2.4 Twin support vector machine (TSVM)

Considering a binary classification task with $m_1$ positive and $m_2$ negative samples where each sample has $n$ dimensions. Let $A \in \mathbb{R}^{m_1 \times n}$ and $B \in \mathbb{R}^{m_2 \times n}$be the matrices holding the vectors of samples $x$ from classes $+1$ and $-1$, respectively.

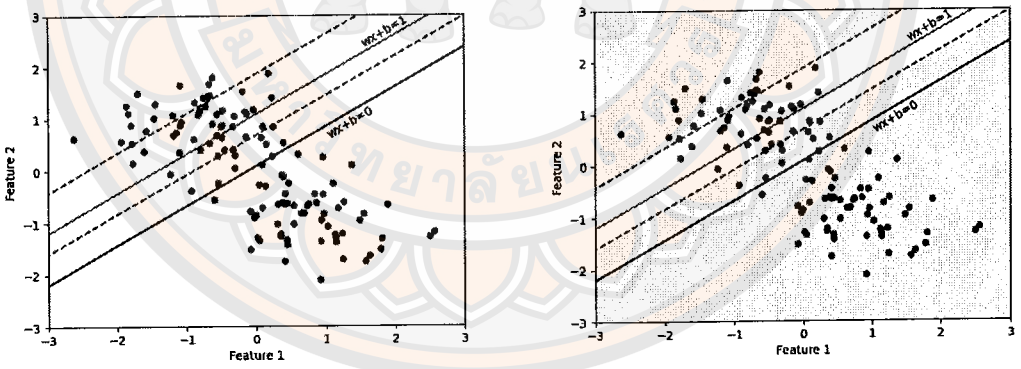



**Figure 8 Geometric interpretation of the concept involves computing the generalized pinball loss.**

According to [3], TSVM generate two non-parallel hyperplanes as shown in Figure 9. The two hyperplanes are specified as follows:

$$\mathbf{w}_1^\top \mathbf{x} + b_1 = 0 \text{ and } \mathbf{w}_2^\top \mathbf{x} + b_2 = 0, \tag{2.4.1}$$

where $\mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^n$ and $b_1, b_2 \in \mathbb{R}$.

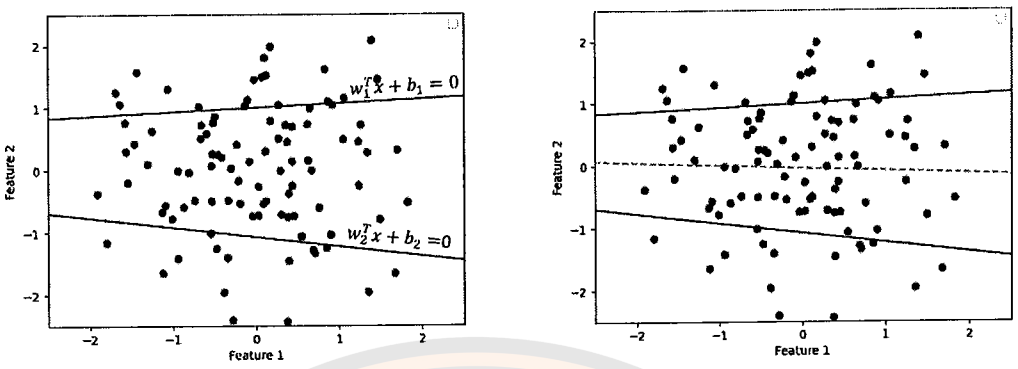



**Figure 9 Illustration of twin support vectors generating nonparallel hyperplanes. The grey dashed line represents the middle between the two hyperplanes.**

The two above hyperplanes are obtained by computing the solutions from the pair of QPPs as follows:

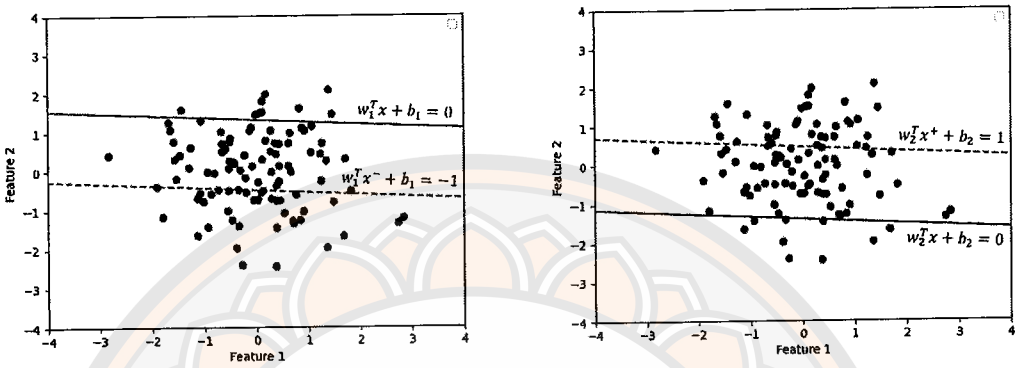
$$\begin{aligned} \min_{\mathbf{w}_1, b_1, \boldsymbol{\xi}} \quad & \frac{1}{2}\|A\mathbf{w}_1 + \mathbf{e}_1 b_1\|^2 + c_1 \mathbf{e}_2^\top \boldsymbol{\xi} \\ \text{s.t.} \quad & -(B\mathbf{w}_1 + \mathbf{e}_2 b_1) + \boldsymbol{\xi} \geq \mathbf{e}_2, \\ & \boldsymbol{\xi} \geq 0, \end{aligned} \tag{2.4.2}$$

and

$$\begin{aligned} \min_{\mathbf{w}_2, b_2, \boldsymbol{\xi}} \quad & \frac{1}{2}\|B\mathbf{w}_2 + \mathbf{e}_2 b_2\|^2 + c_2 \mathbf{e}_1^\top \boldsymbol{\xi} \\ \text{s.t.} \quad & (A\mathbf{w}_2 + \mathbf{e}_1 b_2) + \boldsymbol{\xi} \geq \mathbf{e}_1, \\ & \boldsymbol{\xi} \geq 0, \end{aligned} \tag{2.4.3}$$

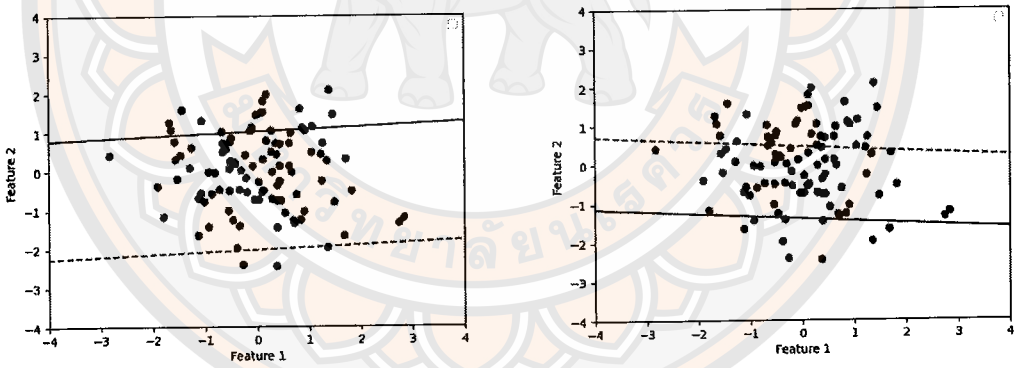
where $\boldsymbol{\xi}$ is slack variable, $c_1, c_2$ are positive penalty parameters, and $\mathbf{e}_1, \mathbf{e}_2$ are vectors of ones with required dimension. For both of the primal problems, an interpretation can be offer in the same way, so only the problem (2.4.2) is consider here. Among the two terms in the objective function, the first term make the positive hyperplane proximal to all positive data, the second term with the two constraints require the positive hyperplane to be at a distance from the negative data by pushing the negative data to the other side of the hyperplane $\mathbf{w}_1^\top \mathbf{x} + b_1 = -1$, where a set $\xi$ of variables is used to measure error whenever the positive

hyperplane is close to the negative data. This interpretation can be state in the Figure 10. Futhermore, the constant $c_1, c_2$ are effect the classification result which also show in the Figure 11.




(a) Consider the positive hyperplane. (b) Consider the negative hyperplane.

**Figure 10 Interpretation of constructing TSVM.**




(a) Positive hyperplane. (b) Negative hyperplane.

**Figure 11 Interpretation of constructing TSVM using different values of $c_1, c_2$.**

Instead of solving the QPPs (2.4.2) and (2.4.3) directly, their dual problems

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^\top G(H^\top H)G^\top \boldsymbol{\alpha} - e_1^\top \boldsymbol{\alpha} \\ \text{s.t.} \quad & \mathbf{0} \leq \boldsymbol{\alpha} \leq c_1 \mathbf{e}_1, \end{aligned} \tag{2.4.4}$$

and

$$\begin{aligned} \min_{\boldsymbol{\beta}} \quad & \frac{1}{2}\boldsymbol{\beta}^\top H(G^\top G)H^\top\boldsymbol{\beta} - e_2^\top\boldsymbol{\beta} \\ \text{s.t.} \quad & \mathbf{0} \leq \boldsymbol{\beta} \leq c_2\mathbf{e}_2, \end{aligned} \tag{2.4.5}$$

are solved, where $G = \begin{bmatrix} B & \mathbf{e}_1 \end{bmatrix}$, $H = \begin{bmatrix} A & \mathbf{e}_2 \end{bmatrix}$ and $\boldsymbol{\alpha}, \boldsymbol{\beta} \geq \mathbf{0}$ are Lagrange multipiers. After getting solutions $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in (2.4.4) and (2.4.5), respectively, the best separating hyperplanes can be obtained by

$$\begin{bmatrix} \mathbf{w}_1 \\ b_1 \end{bmatrix} = -(H^\top H)^{-1}G^\top\boldsymbol{\alpha}, \tag{2.4.6}$$

$$\begin{bmatrix} \mathbf{w}_2 \\ b_2 \end{bmatrix} = (G^\top G)^{-1}H^\top\boldsymbol{\beta}. \tag{2.4.7}$$

A new data $\mathbf{x} \in \mathbb{R}^n$ is imported into (2.4.8) to determine which class it belong to:

$$\text{class}(i) = \arg\min_{i=1,2} \frac{|\mathbf{w}_i^\top\mathbf{x} + b_i|}{\|\mathbf{w}_i\|}, \tag{2.4.8}$$

where $|.|$ is the absolute distance of point $\mathbf{x}$ from the planes $\mathbf{w}_1^\top\mathbf{x} + b_i = 0$, $i = 1, 2$.

In fact, the QPPs of TSVM problems (2.4.2) and (2.4.3), possible to rewrite these as unconstrained optimization problem by the following:

$$\min_{\mathbf{w}_1, b_1, \boldsymbol{\xi}} \quad \frac{1}{2}\|A\mathbf{w}_1 + \mathbf{e}_1 b_1\|^2 + c_1 \sum_{i=1}^{m_2} \mathcal{L}_{hinge}\big(1 + (\mathbf{w}_1^\top\mathbf{x}_i + b_1)\big), \tag{2.4.9}$$

$$\min_{\mathbf{w}_2, b_2, \boldsymbol{\xi}} \quad \frac{1}{2}\|B\mathbf{w}_2 + \mathbf{e}_2 b_2\|^2 + c_2 \sum_{i=1}^{m_1} \mathcal{L}_{hinge}\big(1 - (\mathbf{w}_2^\top\mathbf{x}_i + b_2)\big), \tag{2.4.10}$$

where $\mathcal{L}_{hinge}$ is hinge loss function defined by $\mathcal{L}_{hinge}(v) = \max\{0, v\}$. This loss function is motivated by the idea of the slack variable $\boldsymbol{\xi}$ and already discuss in the previous part. It has resulted in widespread application in classification problems.

Further, considering the terms $(H^\top H)^{-1}$ and $(G^\top G)^{-1}$ in (2.4.6) and (2.4.7), their forms are usually positive semidefinite and cause possible ill-conditioning [28]. So, the approximate inverse matrices of $H^\top H$ and $G^\top G$ are used, which is derived as $H^\top H + \epsilon I$ and $G^\top G + \epsilon I$, respectively, where $\epsilon > 0$ and $I$ is an identity matrix with the required dimensions. However, adding $\epsilon I$ in $H^\top H$ and $G^\top G$, resulting TSVM only get approximate solutions.

To improve this disadvantage, Shao et al. [35] provided a twin bounded support vector machine (TBSVM) by incorporating a regularization term $\frac{1}{2}(\|\mathbf{w}\|^2 + b^2)$ further into TSVM. The problems can be written as follows:

$$\begin{aligned} \min_{\mathbf{w}_1,b_1,\boldsymbol{\xi}} \quad & \frac{1}{2}\|A\mathbf{w}_1 + \mathbf{e}_1 b_1\|^2 + + \frac{1}{2}c_3(\|\mathbf{w}_1\|^2 + b_1^2) + c_1\mathbf{e}_2^\top \boldsymbol{\xi} \\ \text{s.t.} \quad & -(B\mathbf{w}_1 + \mathbf{e}_2 b_1) \geq \mathbf{e}_2 - \boldsymbol{\xi}, \\ & \boldsymbol{\xi} \geq \mathbf{0}, \end{aligned} \tag{2.4.11}$$

and

$$\begin{aligned} \min_{\mathbf{w}_2,b_2,\boldsymbol{\xi}} \quad & \frac{1}{2}\|B\mathbf{w}_2 + \mathbf{e}_2 b_2\|^2 + + \frac{1}{2}c_4(\|\mathbf{w}_2\|^2 + b_2^2) + c_2\mathbf{e}_1^\top \boldsymbol{\xi} \\ \text{s.t.} \quad & A\mathbf{w}_2 + \mathbf{e}_1 b_2 \geq \mathbf{e}_1 - \boldsymbol{\xi}, \\ & \boldsymbol{\xi} \geq \mathbf{0}, \end{aligned} \tag{2.4.12}$$

where $c_1, c_2, c_3, c_4 \geq 0$ are penalty parameters. This model is a robust and effective classifier which has been proposed in a wide variety of versions [29], [30], [31].

## 2.5 Twin support vetor machine with pinball loss (Pin-TSVM)

Tanveer et al. [8] proposed a twin support vector machine using the pinball loss function (Pin-TSVM) to address the shortcomings of hinge loss-based support vector machines. As a result, the model is less noise-sensitive and exhibits stability re-sampling greater than the TSVM. The Pin-TSVM produces a pair of QPPs similar to TSVM, but the constraints are corresponding to pinball loss. Their optimization problems are written as follows:

$$\begin{aligned} \min_{\mathbf{w}_1,b_1,\boldsymbol{\xi}} \quad & \frac{1}{2}\|A\mathbf{w}_1 + \mathbf{e}_1 b_1\|^2 + c_1\mathbf{e}_2^\top \boldsymbol{\xi} \\ \text{s.t.} \quad & -(B\mathbf{w}_1 + \mathbf{e}_2 b_1) \geq \mathbf{e}_2 - \boldsymbol{\xi}, \\ & -(B\mathbf{w}_1 + \mathbf{e}_2 b_1) \leq \mathbf{e}_2 + \frac{\boldsymbol{\xi}}{\tau}, \\ & \boldsymbol{\xi} \geq \mathbf{0}, \end{aligned} \tag{2.5.1}$$

and

$$\min_{\mathbf{w}_2,b_2,\boldsymbol{\xi}} \quad \frac{1}{2}\|B\mathbf{w}_2 + \mathbf{e}_2 b_2\|^2 + c_2\mathbf{e}_1^\top \boldsymbol{\xi}$$

$$\begin{aligned} \text{s.t.} \quad & A\mathbf{w}_2 + \mathbf{e}_1 b_2 \geq \mathbf{e}_1 - \boldsymbol{\xi}, \\ & A\mathbf{w}_2 + \mathbf{e}_1 b_2 \leq \mathbf{e}_1 + \frac{\boldsymbol{\xi}}{\tau}, \\ & \boldsymbol{\xi} \geq \mathbf{0}. \end{aligned} \tag{2.5.2}$$

Here $\boldsymbol{\xi}$ is a slack variable and $c_1, c_2$ are non-negative parameters. The dual problems of (2.5.1) and (2.5.2) can be stated in the following ways:

$$\begin{aligned} \min_{\boldsymbol{\alpha},\boldsymbol{\gamma}} \quad & \frac{1}{2}(\boldsymbol{\alpha} - \boldsymbol{\gamma})^\top G(H^\top H)G^\top(\boldsymbol{\alpha} - \boldsymbol{\gamma}) - \mathbf{e}_2^\top(\boldsymbol{\alpha} - \boldsymbol{\gamma}) \\ \text{s.t.} \quad & \mathbf{0} \leq \boldsymbol{\alpha} + \frac{\boldsymbol{\gamma}}{\tau} \leq c_1\mathbf{e}_2 \\ & \boldsymbol{\alpha} \geq \mathbf{0}, \ \boldsymbol{\gamma} \geq \mathbf{0}, \end{aligned} \tag{2.5.3}$$

and

$$\begin{aligned} \min_{\boldsymbol{\eta},\boldsymbol{\rho}} \quad & \frac{1}{2}(\boldsymbol{\eta} - \boldsymbol{\rho})^\top H(G^\top G)H^\top(\boldsymbol{\eta} - \boldsymbol{\rho}) - \mathbf{e}_1^\top(\boldsymbol{\eta} - \boldsymbol{\rho}) \\ \text{s.t.} \quad & \boldsymbol{\eta} + \frac{\boldsymbol{\rho}}{\tau} \leq c_1\mathbf{e}_1, \\ & \boldsymbol{\eta} \geq \mathbf{0}, \ \boldsymbol{\rho} \geq \mathbf{0}, \end{aligned} \tag{2.5.4}$$

where $G = \begin{bmatrix} B & e_2 \end{bmatrix}$ and $H = \begin{bmatrix} A & e_1 \end{bmatrix}$, $\boldsymbol{\alpha} \geq \mathbf{0}$, $\boldsymbol{\gamma} \geq \mathbf{0}$, $\boldsymbol{\eta} \geq \mathbf{0}$ and $\boldsymbol{\rho} \geq \mathbf{0}$ are Lagrange multipliers. After solving these QPPs (2.5.3) and (2.5.4), then the solution $(w_1, b_1)$ of the primal problem (2.5.1) can be computed as follows:

$$\begin{bmatrix} \mathbf{w}_1 \\ b_1 \end{bmatrix} = -(H^\top H)^{-1}G^\top(\boldsymbol{\alpha} - \boldsymbol{\gamma}). \tag{2.5.5}$$

For the primal problem (2.5.2), we obtain

$$\begin{bmatrix} \mathbf{w}_2 \\ b_2 \end{bmatrix} = (G^\top G)^{-1}H^\top(\boldsymbol{\eta} - \boldsymbol{\rho}). \tag{2.5.6}$$

The two non-parallel hyperplanes are expressed similarly to (2.4.1). A sample point $\mathbf{x} \in \mathbb{R}^n$ is allocated to the positive or negative class by

$$\text{class}(i) = \arg\min_{i=1,2} \frac{|\mathbf{x}^\top \mathbf{w}_i + b_i|}{\|\mathbf{w}_i\|}, \tag{2.5.7}$$

where $|.|$ denotes the absolute distance of point $x$ from the planes $x^\top \mathbf{w}_i + b_i = 0$, $i = 1, 2$ and $(\mathbf{w}_i, b_i)$ is the solution of the problems (2.5.1) and (2.5.2).

## 2.6 Twin support vector machine with generalized pinball loss (GPin-TSVM)

Panup et al. [18] has recently introduced a variant of the TSVM called the generalized pinball loss TSVM, denoted as the GPin-TSVM. The definition of the generalized pinball loss function is given as follows

$$\mathcal{L}^{\epsilon_1,\epsilon_2}_{\tau_1,\tau_2}(v) = \begin{cases} \tau_1\big(v - \frac{\epsilon_1}{\tau_1}\big), & v > \frac{\epsilon_1}{\tau_1}, \\ 0, & -\frac{\epsilon_2}{\tau_2} \leq v \leq \frac{\epsilon_1}{\tau_1}, \\ -\tau_2\big(u + \frac{\epsilon_2}{\tau_2}\big), & v < -\frac{\epsilon_2}{\tau_2}, \end{cases} \tag{2.6.1}$$

where $u = 1 - y(\mathbf{w}^\top \mathbf{x} + b)$ and $\tau_1, \tau_2, \epsilon_1, \epsilon_2 \geq 0$. This improvement from the $\epsilon$-insensitive zone [4] results in enhanced model sparsity while retaining all the inherent properties of the original. The optimization problems are outlined as follows:

$$\begin{aligned} \min_{\mathbf{w}_1, b_1, \boldsymbol{\xi}} \ & \frac{1}{2}\|A\mathbf{w}_1 + \mathbf{e}_1 b_1\|^2 + c_1 \mathbf{e}_2^\top \boldsymbol{\xi} \\ \text{s.t.} \ & -(B\mathbf{w}_1 + \mathbf{e}_2 b_1) \geq \mathbf{e}_2 - \frac{1}{\tau_1}(\boldsymbol{\xi} + \mathbf{e}_2 \epsilon_1), \\ & -(B\mathbf{w}_1 + \mathbf{e}_2 b_1) \leq \mathbf{e}_2 + \frac{1}{\tau_2}(\boldsymbol{\xi} + \mathbf{e}_2 \epsilon_2), \\ & \boldsymbol{\xi} \geq 0, \end{aligned} \tag{2.6.2}$$

and

$$\begin{aligned} \min_{\mathbf{w}_2, b_2, \boldsymbol{\xi}} \ & \frac{1}{2}\|B\mathbf{w}_2 + \mathbf{e}_2 b_2\|^2 + c_2 \mathbf{e}_1^\top \boldsymbol{\xi} \\ \text{s.t.} \ & (A\mathbf{w}_2 + \mathbf{e}_1 b_2) \geq \mathbf{e}_1 - \frac{1}{\tau_3}(\boldsymbol{\xi} + \mathbf{e}_1 \epsilon_3), \\ & (A\mathbf{w}_2 + \mathbf{e}_1 b_2) \leq \mathbf{e}_1 + \frac{1}{\tau_4}(\boldsymbol{\xi} + \mathbf{e}_1 \epsilon_4), \\ & \boldsymbol{\xi} \geq 0. \end{aligned} \tag{2.6.3}$$

Here $\boldsymbol{\xi}$ is a slack variable and $c_1, c_2$ are non-negative parameters. Similarly to the other TSVM, the non-parallel hyperplanes are obtained, and new data are classified by the criterion (2.5.7).

However, the aforemention models are supervised classifiers that perform their training approach entirely with labeled data. When labeled information is

insufficient, the generalization capacity of model frequency worsens. As a result, the semi-supervised classification paradigm, which builds superior classifiers by combining labeled and unlabeled data rather than just labeled data, has gotten much attention. To extend TBSVM to the semi-supervised scenario, Qi et al. [14] introduced an efficient model called Laplacian twin support vector machine (Lap-TSVM). It enhances the classification performance utilizing the graph Laplacian approach to preserve the data manifold structure information. The details of Lap-TSVM are given in the following subsection.

In the context of the semi-supervised framework, we consider a dataset comprising both labeled and unlabeled data denoted as $X_l = \{\mathbf{x}_i\}_{i=1}^{l} \in \mathbb{R}^{l \times n}$, where labels are represented by $Y = \{y_i\}_{i=1}^{l} \in \mathbb{R}^{l}$ with $y_i \in \{+1, -1\}$. The unlabeled data is represented by $X_u = \{\mathbf{x}_i\}_{i=l+1}^{l+u} \in \mathbb{R}^{u \times n}$, where $\mathbf{x}_i \in \mathbb{R}^{n}$. Additionally, we use the matrix $M \in \mathbb{R}^{(l+u) \times n}$ to incorporate both labeled and unlabeled data. The labeled data from the class $+1$ is denoted by $A \in \mathbb{R}^{m_1 \times n}$, and $B \in \mathbb{R}^{m_2 \times n}$ represents data from the class $-1$, where $m_1 + m_2 = l$.

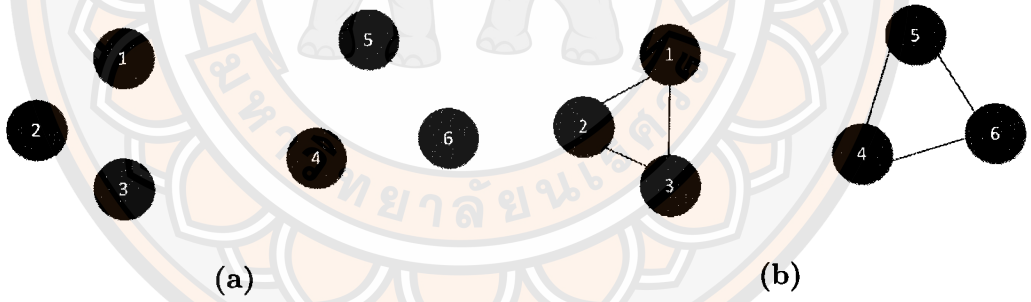



**Figure 12 The circle represent six data point. The line between point exist after using k-nearest neighbor (k=2).**

## 2.7 Laplacian twin support vector machine (Lap-TSVM)

Lap-TSVM aims to enhance the semi-supervised problem, which could be handled by the supervised nonparallel hyperplane classifier TSVM. It is based on hinge loss function and gives the following QPPs:

$$\min_{\mathbf{w}_1, b_1, \boldsymbol{\xi}} \ \frac{1}{2}\|A\mathbf{w}_1 + \mathbf{e}_1 b_1\|^2 + c_1 \mathbf{e}_2^{\top} \boldsymbol{\xi} + \frac{c_2}{2}(\|\mathbf{w}_1\|^2 + b_1^2)$$

$$
\begin{aligned}
&+ \frac{c_3}{2}(M\mathbf{w}_1 + \mathbf{e}b_1)^\top L(M\mathbf{w}_1 + {}_1) \\
\text{s.t.}\ \ &-(B\mathbf{w}_1 + \mathbf{e}_2 b_1) + \boldsymbol{\xi} \geq \mathbf{e}_2, \\
&\boldsymbol{\xi} \geq 0,
\end{aligned}
\tag{2.7.1}
$$

and

$$
\begin{aligned}
\min_{\mathbf{w}_2, b_2, \boldsymbol{\xi}}\ \ &\frac{1}{2}\|B\mathbf{w}_2 + \mathbf{e}_2 b_2\|^2 + c_1 \mathbf{e}_1^\top \boldsymbol{\xi} + \frac{c_2}{2}(\|\mathbf{w}_2\|^2 + b_2^2) \\
&+ \frac{c_3}{2}(M\mathbf{w}_2 + \mathbf{e}b_2)^\top L(M\mathbf{w}_2 + \mathbf{e}b_2) \\
\text{s.t.}\ \ &A\mathbf{w}_2 + \mathbf{e}_1 b_2 + \boldsymbol{\xi} \geq \mathbf{e}_1, \\
&\boldsymbol{\xi} \geq 0,
\end{aligned}
\tag{2.7.2}
$$

where $\boldsymbol{\xi}$ is slack variables, $c_1, c_2$ and $c_3$ are positive penalty parameters, $\mathbf{e}_1, \mathbf{e}_2$ are vectors of ones of appropriate dimensions.

Here, $M$ is the matrix represents both labeled and unlabeled data. $L$ denotes the graph Laplacian defined as: $L = D - A$, where $D$ is the diagonal matrix of vertex degrees, $A$ is the adjacency matrix of the graph derived from weight matrix defined by k-nearest neighbors, expressed as follows:

$$
W_{ij} = \begin{cases} \exp(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}), & \text{if } \mathbf{x}_i, \mathbf{x}_j \text{ are neighbor;} \\ 0, & \text{otherwise.} \end{cases}
\tag{2.7.3}
$$

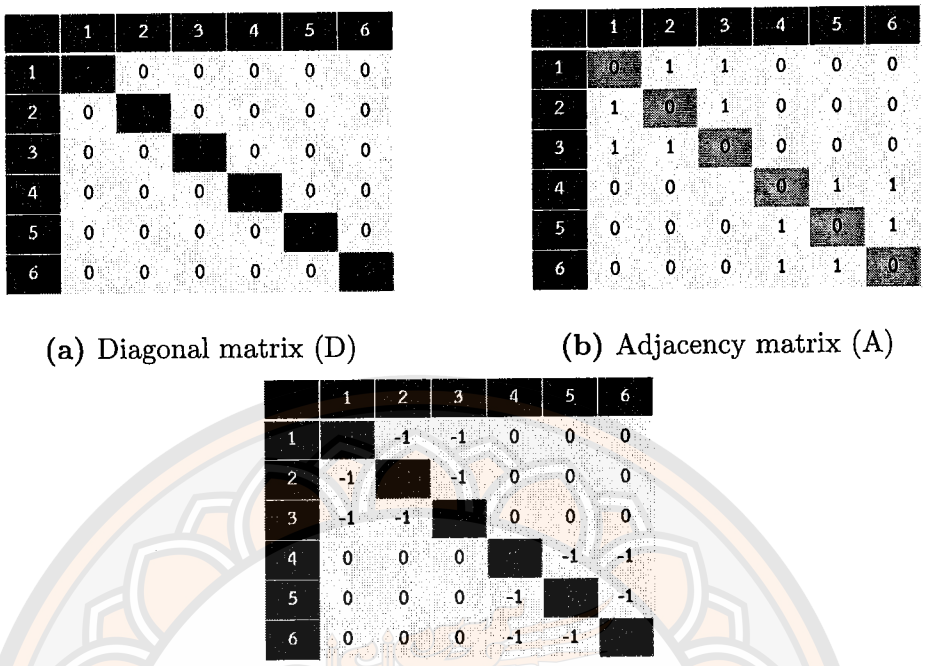
The process of computing the graph Laplacian is depicted in Figures 12 and 13, respectively. Lap-TSVM finds a pair of nonparallel-planes as follows:

$$
\mathbf{x}^\top \mathbf{w}_1 + b_1 = 0 \text{ and } \mathbf{x}^\top \mathbf{w}_2 + b_2 = 0,
\tag{2.7.4}
$$

where $\mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^n$ and $b_1, b_2 \in \mathbb{R}$. The class prediction of a new sample $\mathbf{x} \in \mathbb{R}^n$ is determined by which of the two hyperplanes (2.7.4) it is closest to, i.e.

$$
\text{class}(i) = \arg\min_{i=1,2} \frac{|\mathbf{x}^\top \mathbf{w}_i + b^{(i)}|}{\|\mathbf{w}_i\|},
\tag{2.7.5}
$$

where $|.|$ denotes the absolute distance of point $x$ from the planes $\mathbf{x}^\top \mathbf{w}_i + b_i = 0$. Several classification results obtained demonstrate the performance of Lap-TSVM, as illustrated in Figure 14.



| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | |

(a) Diagonal matrix (D)

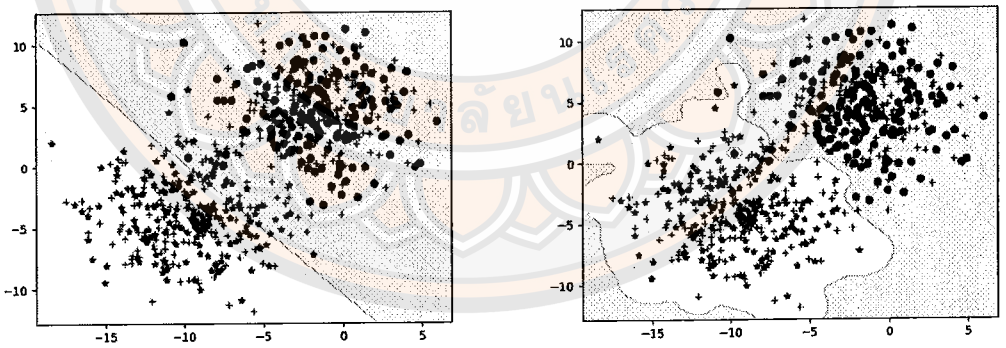
| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 2 | 1 | 0 | 1 | 0 | 0 | 0 |
| 3 | 1 | 1 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 1 | 1 |
| 5 | 0 | 0 | 0 | 1 | 0 | 1 |
| 6 | 0 | 0 | 0 | 1 | 1 | 0 |

(b) Adjacency matrix (A)

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | | -1 | -1 | 0 | 0 | 0 |
| 2 | -1 | | -1 | 0 | 0 | 0 |
| 3 | -1 | -1 | | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | | -1 | -1 |
| 5 | 0 | 0 | 0 | -1 | | -1 |
| 6 | 0 | 0 | 0 | -1 | -1 | |

(c) Laplacian matrix (L)

**Figure 13 Illustration of computation Laplacian matrix ($L = D - A$).**




(a) Linear kernel

(b) RBF kernel

**Figure 14 Some classification results obtained by Lap-TSVM using the linear kernel. The red circle and green star represent the positive and negative labeled data, respectively. The + denotes the unlabeled data for each class. Some classification results obtained by Lap-TSVM using the RBF kernel.**

The models we discuss above are built by learning from exact data input values, assuming that a singular value can represent each feature of the data samples. However, uncertainties may arise in real-world scenarios due to measurement and statistical errors during data collection, affecting the samples. Consequently, there have been studies for alternative methods to effectively handle uncertain data modifications in such cases.

## 2.8 Random variable and Guassian distribution

**Definition 2.8.1.** [27] A **sample space** $\Omega$ is the set of all possible outcomes from an experiment. We denote $\xi$ as an element in $\Omega$.

**Definition 2.8.2.** [27] A **random variable** $X$ is a function $X : \Omega \to \mathbb{R}$ that map the outcome $\xi \in \Omega$ to a number $X(\xi)$ on the real line.

Random variables can be classified into two main types: Discrete Random Variable: This type of random variable can only take on distinct, separate values. Continuous Random Variable: This type of random variable can take on any value within a certain range.

In this study, our attention is directed toward the continuous random variable, and the concepts we employ for the work are as follows.

**Definition 2.8.3.** [27] The expectation of a continuous random variable $X$ is

$$E[X] = \int_{\Omega} \mathbf{x} f_X(x) d\mathbf{x}. \tag{2.8.1}$$

If a function $g$ is applied to the random variable $X$, the expectation can be found using the following theorem.

**Theorem 2.8.4.** [27] Let $g : \Omega \to \mathbb{R}$ be a function and $X$ be a continuous random variable. Then

$$E[g(X)] = \int_{\Omega} g(\mathbf{x}) f_X(\mathbf{x}) d\mathbf{x}. \tag{2.8.2}$$

**Definition 2.8.5.** [27] For a univariate random variable, the Guassian distribution has a density that is given by

$$f_X(\mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left( - \frac{(\mathbf{x} - \mu)^2}{2\sigma^2} \right) \tag{2.8.3}$$

where $(\mu, \sigma^2)$ are mean and variance which are parameters of the distribution.

**Definition 2.8.6.** [27] The Gaussian distribution with $\mu = 0$ and $\sigma^2 = 1$ is referred to as the Standard normal distribution and has a PDF

$$f_X(\mathbf{x}) = \frac{1}{\sqrt{2\pi}} \exp\left( -\frac{\mathbf{x}^2}{2} \right) \tag{2.8.4}$$

The CDF of the standard Gaussian can be determined by the integrating the PDF. We have a special notation for this CDF.

**Definition 2.8.7.** [27] The CDF of the standard Gaussian is defined as the $\Phi(\cdot)$ function

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\mathbf{x}} e^{-\frac{t^2}{2}} dt. \tag{2.8.5}$$

The CDF of standard Gaussian is related to a so-called error function defined as

$$\text{efr} = \frac{2}{\pi} \int_0^{\mathbf{x}} e^{-t^2} dt \tag{2.8.6}$$

For multivariate Guassian distribution, the definition is fully characterized by the mean vector $\boldsymbol{\mu}$ and a covariance matrix $\Sigma$ and defined as

$$f_X(\mathbf{x}) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left( -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right) \tag{2.8.7}$$

The special case of Guassian with zero mean and identity covariance, that is, $\boldsymbol{\mu} = 0$ and $\Sigma = I$, is referred to as the standard normal distribution.

**Lemma 2.8.8.** [23] Assume that $Z$ is a one-dimensional Gaussian random variable with the mean $\mu$ and standard deviation $\sigma$. The integrals are defined as

$$I_1(\mu, \sigma) = \int_{z \geq a} z f_Z(z) dz$$
$$I_2(\mu, \sigma) = \int_{z \leq a} z f_Z(z) dz$$

Then $I_1(\mu, \sigma)$ and $I_2(\mu, \sigma)$ can be expressed as follows in terms of the complementary error function

$$I_1(\mu, \sigma) = \frac{\mu}{2}\left(1 - \text{erf}\left( \frac{a - \mu}{\sqrt{2}\sigma} \right)\right) + \frac{\sigma}{\sqrt{2\pi}} \exp\left( -\frac{\mu^2}{2\sigma^2} \right),$$
$$I_2(\mu, \sigma) = \frac{\mu}{2}\left(1 - \text{erf}\left( -\frac{a - \mu}{\sqrt{2}\sigma} \right)\right) - \frac{\sigma}{\sqrt{2\pi}} \exp\left( -\frac{\mu^2}{2\sigma^2} \right),$$

where the complementary error function $erf(\mathbf{x}) = \frac{2}{\sqrt{\pi}} \int_0^{\mathbf{x}} e^{-t^2} dt$ [33].

**Lemma 2.8.9.** [23] Assume that $X$ is an n-dimensional Guassian random vector with the mean $U$ and the covariance matrix $\Sigma$. The affine function $\mathbf{w}^\top X + b$ is a Guassian random variable, the mean of this function is $\mathbf{w}^\top U + b$, and the variance of this function is $\mathbf{w}^\top \Sigma \mathbf{w}$

## 2.9 Support vector machines with $\epsilon$-insensitive pinball loss for uncertain data (UPinSVMs)

Consider uncertainty of the training samples, we give a multivariate Gaussian distribution with mean vector $\mathbf{x}_i \in \mathbb{R}^n$ and covariance matrix $\Sigma_i \in \mathbb{R}^{n\times n}$ $(i = 1, \ldots, m)$ as $(\mathbf{x}_1, \Sigma_1, y_1), (\mathbf{x}_2, \Sigma_2, y_2), \ldots, (\mathbf{x}_m, \Sigma_m, y_m)$, where $y_i \in \{1, -1\}$. Then, we define $m$ random variables, $X_i$, each of which we assume that follows the corresponding $n$-dimensional Gaussian distribution $N(\mathbf{x}_i, \Sigma_i)$.

To enhance the effectiveness of the model, Liang et al. [23] introduced SVMs with an $\epsilon$-insensitive pinball loss function specifically designed for uncertain data. In this context, the data is considered as an uncertain sample involving a random vector $X_i$ with $n$-dimensional Gaussian distribution $N(\mathbf{x}_i, \Sigma_i)$. By formulating the optimization problem with the expectation of loss, the following approach was developed.

$$\min_{\mathbf{w},b} \frac{1}{2}(\|\mathbf{w}\|^2 + b^2) + C\sum_{i=1}^{m} \int_{\mathbb{R}^n} \mathcal{L}^{\epsilon}_{pin}\big(1 - y_i(\mathbf{w}^\top \mathbf{x} + b)\big) f_{X_i}(\mathbf{x}) d\mathbf{x}. \tag{2.9.1}$$

The second term in (2.9.1) can be represented in term of the complementary error function as follows:

$$\begin{aligned} L_i(\mathbf{w}, b) = {} & \frac{d_i^1}{2}\big(1 - \operatorname{erf}(f_i^1)\big) + \frac{e_i}{2\sqrt{\pi}} \exp\left( - \left(f_i^1\right)^2 \right) \\ & + \frac{d_i^2}{2}\big(1 - \operatorname{erf}(f_i^2)\big) + \frac{e_i}{2\sqrt{\pi}} \exp\left( - \left(f_i^2\right)^2 \right), \end{aligned} \tag{2.9.2}$$

where $d_i^1 = 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b) - \epsilon$, $d_i^2 = -\tau\big(1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)\big) - \epsilon$, $e_i = \sqrt{2\mathbf{w}^\top \Sigma_i \mathbf{w}}$, $f_i^1 = -\frac{d_i^1}{e_i}$, and $f_i^2 = -\frac{d_i^2}{e_i}$. For simplicity, they reformulated the optimization problem of (2.9.1) by using the complimentary error function in (2.9.2) as follows.

$$\min_{\mathbf{w},b} h(\mathbf{w}, b) = C\sum_{i=1}^{m} L_i(\mathbf{w}, b) + \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{2}b^2, \tag{2.9.3}$$

where $L_i(\mathbf{w}, b)$ is defined in (2.9.2). Since the objective function of (2.9.3) is strongly convex with $\mathbf{w}$ and $b$, the unique solution are obtained.

## 2.10 Statistical analysis

To justify the competitive performance of the experimental results, statistical analysis is employed. In this research, nonparametric statistics are considered. The most well-known procedure for testing the differences between more than two related samples is the Friedman test. The related samples in classification are the performances of the methods measured across the same data sets. It considers that the null hypothesis being tested is that all methods obtain similar results with nonsignificant differences.

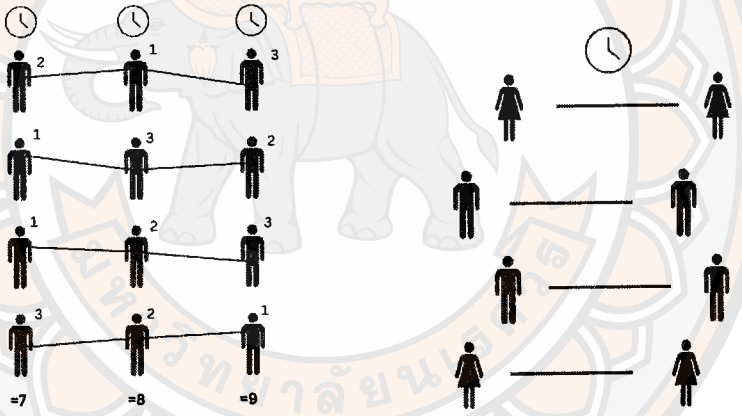



**Figure 15 Fried-men test and Wicoxon sign rank test.**

The Friedman test [34] serves as a nonparametric alternative to the parametric two-way analysis of variance. Its aim is to determine, based on a sample of results, whether differences exist among treatment effects. In the initial stage of computing the test statistic, the original results are converted into ranks. Consequently, the algorithms are ranked independently for each problem. The ranks are assigned by ordering the values either in ascending or descending order. In descending order, values are arranged from the highest to the lowest, with the highest values assigned the rank of 1, and so on. Therefore, a lower average rank generally

indicates better performance. The Friedman test is computed from

$$\mathcal{X}_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right],$$
$$F_F = \frac{(N-1)\mathcal{X}_F^2}{N(k-1)-\mathcal{X}_F^2}.$$

Here, $N$ number of datasets, $k$ is the number of algorithms, $R_j = \frac{1}{N}\sum_{j=1}^{N} r_j$, and $F_F$ is $F$-distribution with a degree of freedom $(k-1)(N-1)$. If the null hypothesis is rejected, a post hoc test can be conducted. In this context, the Nemenyi post hoc test is utilized, and it is computed from the critical difference (CD) value, which has the formula $CD = q_\alpha\sqrt{\frac{k(k+1)}{6N}}$, where $q_\alpha$ is the critical value for a chosen significance level (e.g., 0.05),$k$ is the number of groups being compared, and $N$ is the total number of observations. In summary, the critical values from the Nemenyi test are involved in determining the Critical Difference, which is used to evaluate the significance of pairwise differences between groups. If the absolute value of the pairwise difference exceeds the Critical Difference, it suggests a statistically significant distinction between those groups. The differences between the two methods are shown in the Figure 15.

The next part will discuss another tool used in research, specifically the Wilcoxon sign rank test. It assess whether there is a significant difference between two related groups.The null hypothesis is that the median of the paired differences between two related groups is zero. If the $p$-value associated with the test falls below a chosen significance level (e.g., 0.05), the null hypothesis is rejected, indicating evidence of a significant difference between the medians of the paired groups.

Additionally, the analysis also involves counting the number of Win/-Draw/Loss to assess the experimental outcomes. A win is defined as having the highest accuracy compared to the other methods, a loss as not having the highest accuracy, and a draw occurs when two methods exactly attain the highest accuracy.

## 2.11 Evaluation metrics

In machine learning, evaluation metrics are essential for quantitatively assessing the performance of models. Several key metrics are commonly used for

this purpose:

1. **Accuracy**: This metric measures the proportion of correctly classified instances out of the total instances. It is calculated as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

where $TP$ (True Positive) is the number of correctly predicted positive instances. $TN$ (True Negative) is the number of correctly predicted negative instances. $FP$ (False Positive) is the number of incorrectly predicted positive instances. $FN$ (False Negative) is the number of incorrectly predicted negative instances.

2. **Precision**: Precision quantifies the accuracy of positive predictions. It is the ratio of true positive instances to the total predicted positive instances, calculated as:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Precision focuses on the accuracy of positive predictions and is useful when the cost of false positives is high.

3. **Recall (Sensitivity)**: Recall evaluates the ability of the model to correctly identify all positive instances. It is the ratio of true positive instances to the total actual positive instances, given by:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Recall is valuable when the cost of false negatives is high, as it indicates the percentage of actual positive cases correctly identified by the model.

4. **F1-score**: The F1-score is the harmonic mean of precision and recall, providing a balanced measure of a classifier's performance. It is calculated as:

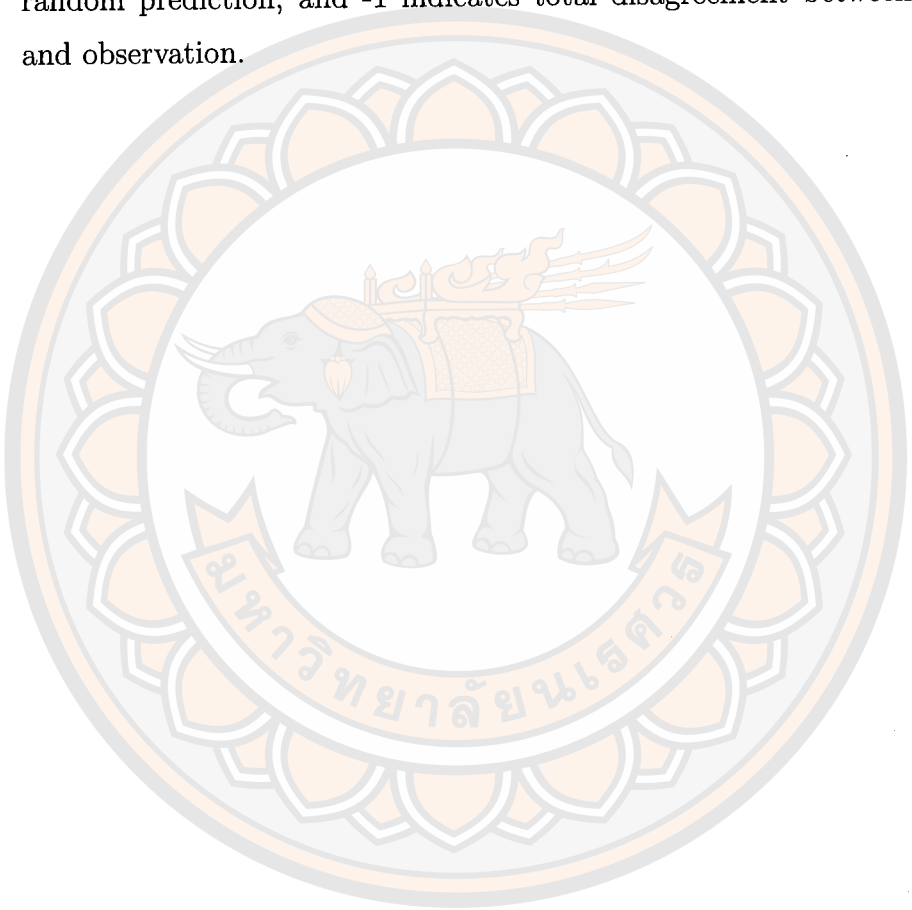
$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

F1-score ranges from 0 to 1, where 1 indicates perfect precision and recall, and 0 indicates poor performance.

5. **Matthews Correlation Coefficient (MCC)**: MCC considers both true and false positives and negatives, making it useful for binary classifications. It ranges from -1 to 1 and is calculated as:

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

MCC ranges from -1 to 1, where 1 indicates perfect prediction, 0 indicates random prediction, and -1 indicates total disagreement between prediction and observation.



# CHAPTER III

# LAPLACIAN TWIN SUPPORT VECTOR MACHINE FOR SEMI-SUPERVISED CLASSIFICATION

## 3.1 Laplacian twin support vector machine with pinball loss for semi-supervised classification

We present a model called Laplacian twin support vector machine with pinball loss (Lap-PTSVM) which constructs nonparallel-hyperplanes and utilize the concept of quantile distance of pinball loss to measure an error. The motivation for this improvement is according to the Lap-TSVM making the model unable to deal with the noise problem of the data. Our proposed inherits many properties, including noise insensitivity, stability for resampling, and also can be analyzed the scatter minimization. In both linear and non-linear cases, consideration and explanation are as follows:

### 3.1.1 Linear Case

We provide the Lap-PTSVM classifier, which brings better noise insensitivity and is stable for resampling over the original Lap-TSVM. Also, it gives the inherited property of scatter minimization. In the standard Lap-TSVM, we apply the pinball loss and generate the following QPPs:

$$\min_{\mathbf{w}_1,b_1} \frac{1}{2}\|A\mathbf{w}_1 + \mathbf{e}_1 b_1\|^2 + \frac{c_2}{2}(\|\mathbf{w}_1\|^2 + b_1^2) + \frac{c_3}{2}(M\mathbf{w}_1 + \mathbf{e}b_1)^\top L(M\mathbf{w}_1 + \mathbf{e}b_1) + c_1\mathbf{e}_2^\top \mathcal{L}_\tau\big(\mathbf{e}_2 + (B\mathbf{w}_1 + \mathbf{e}_2 b_1)\big), \tag{3.1.1}$$

and

$$\min_{\mathbf{w}_2,b_2} \frac{1}{2}\|B\mathbf{w}_2 + \mathbf{e}_2 b_2\|^2 + \frac{c_2}{2}(\|\mathbf{w}_2\|^2 + b_2^2) + \frac{c_3}{2}(M\mathbf{w}_2 + \mathbf{e}b_2)^\top L(M\mathbf{w}_2 + \mathbf{e}b_2) + c_1\mathbf{e}_1^\top \mathcal{L}_\tau\big(\mathbf{e}_1 - (A\mathbf{w}_2 + \mathbf{e}_1 b_2)\big), \tag{3.1.2}$$

where $\mathcal{L}_\tau$ represents the pinball loss function and The formula of Laplacian matrix graph $L$ is similar to Lap-TSVM. We can convert the problems (3.1.1) and (3.1.2)

into the equivalent familiar formulations (2.5.1) and (2.5.2) with the apply of a slack variable $\xi$, then we obtain the following QPPs:

$$\begin{aligned}\min_{\mathbf{w}_1,b_1,\boldsymbol{\xi}} \ & \frac{1}{2}\|A\mathbf{w}_1+\mathbf{e}_1b_1\|^2+c_1\mathbf{e}_2^\top\boldsymbol{\xi}+\frac{c_2}{2}(\|\mathbf{w}_1\|^2+b_1^2)\\ & +\frac{c_3}{2}(M\mathbf{w}_1+\mathbf{e}b_1)^\top L(M\mathbf{w}_1+\mathbf{e}b_1)\\ \text{s.t.} \ & -(B\mathbf{w}_1+\mathbf{e}_2b_1)\geq \mathbf{e}_2-\boldsymbol{\xi},\\ & -(B\mathbf{w}_1+\mathbf{e}_2b_1)\leq \mathbf{e}_2+\frac{\boldsymbol{\xi}}{\tau},\\ & \boldsymbol{\xi}\geq 0,\end{aligned} \tag{3.1.3}$$

and

$$\begin{aligned}\min_{\mathbf{w}_2,b_2,\boldsymbol{\xi}} \ & \frac{1}{2}\|B\mathbf{w}_2+\mathbf{e}_2b_2\|^2+c_1\mathbf{e}_1^\top\boldsymbol{\xi}+\frac{c_2}{2}(\|\mathbf{w}_2\|^2+b_2^2)\\ & +\frac{c_3}{2}(M\mathbf{w}_2+\mathbf{e}b_2)^\top L(M\mathbf{w}_2+\mathbf{e}b_2)\\ \text{s.t.} \ & A\mathbf{w}_2+\mathbf{e}_1b_2\geq \mathbf{e}_1-\boldsymbol{\xi},\\ & A\mathbf{w}_2+\mathbf{e}_1b_2\leq \mathbf{e}_1+\frac{\boldsymbol{\xi}}{\tau},\\ & \boldsymbol{\xi}\geq 0,\end{aligned} \tag{3.1.4}$$

where $\tau$ is non-negative real value. To obtain the solution of (3.1.3) and (3.1.4), we convert them to the dual form. Since, they have similar analysis, we only take the problem (3.1.3) into account and introduce the Lagrange multipliers $\boldsymbol{\alpha}\geq\mathbf{0},\boldsymbol{\beta}\geq\mathbf{0}$ and $\boldsymbol{\gamma}\geq\mathbf{0}$, then we get the Lagrangian function corresponding to Lap-PTSVM (3.1.3) as follows:

$$\begin{aligned}L(\mathbf{w}_1,b_1,\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}) =& \frac{1}{2}\|A\mathbf{w}_1+\mathbf{e}_1b_1\|^2+c_1\mathbf{e}_2^\top\boldsymbol{\xi}\\ & +\frac{c_2}{2}(\|\mathbf{w}_1\|^2+b_1^2)\\ & +\frac{c_3}{2}(M\mathbf{w}_1+\mathbf{e}b_1)^\top L(M\mathbf{w}_1+\mathbf{e}b_1)\\ & -\boldsymbol{\alpha}^\top\big(-(B\mathbf{w}_1+\mathbf{e}_2b_1)-\mathbf{e}_2+\boldsymbol{\xi}\big)\\ & -\boldsymbol{\gamma}^\top\big((B\mathbf{w}_1+\mathbf{e}_2b_1)+\mathbf{e}_2+\frac{\boldsymbol{\xi}}{\tau}\big)\\ & -\boldsymbol{\beta}^\top\boldsymbol{\xi}.\end{aligned} \tag{3.1.5}$$

According to the Karush–Kuhn–Tucker (K.K.T.) optimality conditions, we get:

$$\frac{\partial L}{\partial \mathbf{w}_1}=A^\top(A\mathbf{w}_1+\mathbf{e}_1b_1)+c_2\mathbf{w}_1+c_3M^\top L(M\mathbf{w}_1+\mathbf{e}b_1)$$

$$+ B^\top \boldsymbol{\alpha} - B^\top \boldsymbol{\gamma} = 0, \tag{3.1.6}$$

$$\frac{\partial L}{\partial b_1} = \mathbf{e}_1^\top (A\mathbf{w}_1 + \mathbf{e}_1 b_1) + c_2 b_1 + c_3 \mathbf{e}^\top L(M\mathbf{w}_1 + \mathbf{e}b_1)$$
$$+ \mathbf{e}_2^\top \boldsymbol{\alpha} - \mathbf{e}_2^\top \boldsymbol{\gamma} = 0, \tag{3.1.7}$$

$$\frac{\partial L}{\partial \boldsymbol{\xi}} = c_1 \mathbf{e}_2 - \boldsymbol{\alpha} - \boldsymbol{\beta} - \frac{\boldsymbol{\gamma}}{\tau} = 0, \tag{3.1.8}$$

$$\boldsymbol{\alpha}^\top \big( - (B\mathbf{w}_1 + \mathbf{e}_2 b_1) - \mathbf{e}_2 + \boldsymbol{\xi} \big) = 0, \tag{3.1.9}$$

$$\boldsymbol{\beta}^\top \boldsymbol{\xi} = 0, \tag{3.1.10}$$

$$\boldsymbol{\gamma}^\top \big( (B\mathbf{w}_1 + \mathbf{e}_2 b_1) + \mathbf{e}_2 + \frac{\boldsymbol{\xi}}{\tau} \big) = 0. \tag{3.1.11}$$

By using the equation (3.1.8) and $\boldsymbol{\beta} \geq 0$, it implies that

$$\boldsymbol{\alpha} + \frac{\boldsymbol{\gamma}}{\tau} \leq c_1 \mathbf{e}_2. \tag{3.1.12}$$

Adding equation (3.1.6) and (3.1.7) yields:

$$\begin{bmatrix} A^\top \\ \mathbf{e}_1^\top \end{bmatrix} \begin{bmatrix} A & \mathbf{e}_1 \end{bmatrix} \begin{bmatrix} \mathbf{w}_1 \\ b_1 \end{bmatrix} + c_2 \begin{bmatrix} \mathbf{w}_1 \\ b_1 \end{bmatrix} + c_3 \begin{bmatrix} M^\top \\ \mathbf{e}^\top \end{bmatrix} L \begin{bmatrix} M & \mathbf{e} \end{bmatrix} \begin{bmatrix} \mathbf{w}_1 \\ b_1 \end{bmatrix}$$
$$+ \begin{bmatrix} B^\top \\ \mathbf{e}_2^\top \end{bmatrix} (\boldsymbol{\alpha} - \boldsymbol{\gamma}) = 0. \tag{3.1.13}$$

Define $\boldsymbol{\omega} = \boldsymbol{\alpha} - \boldsymbol{\gamma}$, $\boldsymbol{\mu} = \boldsymbol{\eta} - \boldsymbol{\rho}$, $H = \begin{bmatrix} A & \mathbf{e}_1 \end{bmatrix}$, $F = \begin{bmatrix} B & \mathbf{e}_2 \end{bmatrix}$, $J = \begin{bmatrix} M & \mathbf{e} \end{bmatrix}$ and $z_1 = \begin{bmatrix} \mathbf{w}_1 \\ b_1 \end{bmatrix}$. With these notations, the equation (3.1.13) can be rewritten as follows:

$$(H^\top H + c_2 I + c_3 J^\top L J) z_1 + F^\top \boldsymbol{\omega} = 0. \tag{3.1.14}$$

Therefore, equation (3.1.14) becomes the modified equation:

$$z_1 = -(H^\top H + c_2 I + c_3 J^\top L J)^{-1} F^\top \boldsymbol{\omega}. \tag{3.1.15}$$

Using equation (3.1.5) and the stated K.K.T. conditions, we derive the dual form of (3.1.3) as follows:

$$\min_{\boldsymbol{\omega}} \ \frac{1}{2} \boldsymbol{\omega}^\top F (H^\top H + c_2 I + c_3 J^\top L J)^{-1} F^\top \boldsymbol{\omega} - \boldsymbol{\omega}^\top \mathbf{e}_2$$
$$\text{s.t.} \ \ \boldsymbol{\omega} + \boldsymbol{\gamma} \left( 1 + \frac{1}{\tau} \right) \leq c_1 \mathbf{e}_2, \tag{3.1.16}$$

$$\boldsymbol{\omega} + \boldsymbol{\gamma} \geq 0, \boldsymbol{\gamma} \geq 0.$$

Similarly, the dual problem of (3.1.4) can be obtained as follows:

$$\begin{aligned} \min_{\boldsymbol{\mu}} \ & \frac{1}{2}\boldsymbol{\mu}^\top H(F^\top F + c_2 I + c_3 J^\top L J)^{-1} H^\top \boldsymbol{\mu} - \boldsymbol{\mu}^\top \mathbf{e}_1 \\ \text{s.t.} \ & \boldsymbol{\mu} + \boldsymbol{\rho}\left(1 + \frac{1}{\tau}\right) \leq c_1 \mathbf{e}_1, \\ & \boldsymbol{\mu} + \boldsymbol{\rho} \geq 0, \boldsymbol{\rho} \geq 0. \end{aligned} \tag{3.1.17}$$

Finally, the optimal hyperplanes are given by

$$z_1 = \begin{bmatrix} \mathbf{w}_1 \\ b_1 \end{bmatrix} = -(H^\top H + c_2 I + c_3 J^\top L J)^{-1} F^\top \boldsymbol{\omega}^*, \tag{3.1.18}$$

and

$$z_2 = \begin{bmatrix} \mathbf{w}_2 \\ b_2 \end{bmatrix} = (F^\top F + c_2 I + c_3 J^\top L J)^{-1} H^\top \boldsymbol{\mu}^*, \tag{3.1.19}$$

where $\boldsymbol{\omega}^*$ and $\boldsymbol{\mu}^*$ are obtained by solving the problems (3.1.16) and (3.1.17), respectively.

Additionally, we also provide the Python code used in this work. However, the first step is to set up the module we need to use here. Please be aware that all the code presented here originates from the Class (BaseEstimator), resulting in the presence of self in the function input. If you intend to employ the code, we recommend reviewing it once more. All the Python code in this thesis uses the following environment: pandas 1.3.2, numpy 1.20.3, matplotlib 3.5.3, scikit-learn 0.24.2, scipy 1.6.2, and cvxopt 1.3.0.

```
import math
import numpy as np
from sklearn.base import BaseEstimator
from cvxopt import matrix, solvers
from sklearn.model_selection import train_test_split
from sklearn.neighbors import kneighbors_graph
from scipy.sparse import csc_matrix, csr_matrix, isspmatrix,
    dok_matrix, lil_matrix, bsr_matrix
from sklearn.base import BaseEstimator
```

In the upcoming part, we will explore the process of generating the Laplacian matrix. For this task, we employ the following Python code to facilitate the creation of this crucial mathematical construct.

```
def construct_laplacian(self, X):
    knn_dist_graph = kneighbors_graph(X=X,
                                      n_neighbors=self.k,
                                      mode='distance',
                                      metric='euclidean',
                                      n_jobs=6)
    similarity_graph = lil_matrix(knn_dist_graph.shape)
    nonzeroindices = knn_dist_graph.nonzero()
    similarity_graph[nonzeroindices] = np.exp(-np.asarray(
        knn_dist_graph[nonzeroindices])**2 / 2.0 * self.sigma
        **2)
    similarity_graph = 0.5 * (similarity_graph +
        similarity_graph.T)
    degree_matrix = similarity_graph.sum(axis=1)
    diagonal_matrix = np.diag(np.asarray(degree_matrix).reshape
        (X.shape[0],))
    L =  diagonal_matrix - similarity_graph
    L = np.asarray(L)
    return L
```

To obtain the solution, we use the package cvxopt. Consequently, we must convert the problems mentioned in 3.1.16 and 3.1.17 into a format compatible with the cvxopt package. The following code comprises the complete setup and execution of the entire process until the solutions are obtained.

```
def fit(self,M,p):
    #using construct_laplacian function
    L = self.construct_laplacian(M)
    #set percent of unlabeled data
    percent_unlabel = 0.5
    match_pos = (p == 1)
    match_neg = (p == -1)
    n_label_pos = math.floor(match_pos.sum() - (percent_unlabel
        *match_pos.sum()))
    n_label_neg = math.floor(match_neg.sum() - (percent_unlabel
        *match_neg.sum()))
```

```
10          total = M.shape[0]
11          #obtain the positive and negative data
12          A = M[0:n_label_pos,:]
13          B = M[n_label_pos: n_label_neg + n_label_pos  ,:]
14          #consider the positive hyperplane
15          n_label_pos = A.shape[0]
16          n_label_neg = B.shape[0]
17          total = M.shape[0]
18          #construct matrix H (positive data)
19          e1 = matrix(np.ones((n_label_pos, 1)))
20          H = matrix(np.concatenate((A, e1), axis=1))
21          #construct matrix F (negative data)
22          e2 = matrix(np.ones((n_label_neg, 1)))
23          F = matrix(np.concatenate((B, e2), axis=1))
24          #construct matrix J (both label and unlabeled data)
25          t = matrix(np.ones((total, 1)))
26          J = matrix(np.concatenate((M, t), axis=1))
27          #construct (H.TH+c_2I+c_3J.TLJ)^-1
28          s1 = matrix(np.dot(H.T,H) + self.c_2*(np.identity(np.size(H
                ,1))) + self.c_3*np.dot(np.dot(J.T,L),J))
29          d1 = matrix(np.linalg.inv(s1))
30          ' Set the problem into the form of cvxopt '
31          #construct P
32          P1_11 = matrix(np.dot(np.dot(F,d1),F.T))
33          P1_12 = matrix(np.zeros((n_label_neg,n_label_neg)))
34          P1_21 = matrix(np.zeros((n_label_neg,n_label_neg)))
35          P1_22 = matrix(np.zeros((n_label_neg ,n_label_neg)))
36          P1_r1 = matrix(np.concatenate((P1_11, P1_12), axis=1))
37          P1_r2 = matrix(np.concatenate((P1_21, P1_22), axis=1))
38          P1 = matrix(np.concatenate((P1_r1, P1_r2)))
39          #construct q
40          q1_11 = (-1)*matrix(np.ones((n_label_neg,1)))
41          q1_12 = matrix(np.zeros((n_label_neg,1)))
42          q1 = matrix(np.concatenate((q1_11, q1_12)))
43          #construct G
44          G1_11 = (-1/self.tau_1)*matrix(np.identity(n_label_neg))
45          G1_12 = (1+1/self.tau_1)*matrix(np.identity(n_label_neg))
46          G1_21 = matrix(np.zeros((n_label_neg,n_label_neg )))
47          G1_22 = (-1)*matrix(np.identity(n_label_neg))
48          G1_31 = matrix(np.identity(n_label_neg))
```

```
        G1_32 = (-1)*matrix(np.identity(n_label_neg))

        G1_r1 = matrix(np.concatenate((G1_11, G1_12), axis=1))
        G1_r2 = matrix(np.concatenate((G1_21, G1_22), axis=1))
        G1_r3 = matrix(np.concatenate((G1_31, G1_32), axis=1))
        G1 = matrix(np.concatenate((G1_r1, G1_r2, G1_r3)))
        #construct h
        h1_31 = self.c_1*matrix(np.ones((n_label_neg,1)))
        h1_32 = matrix(np.zeros((n_label_neg,1)))
        h1_33 = matrix(np.zeros((n_label_neg ,1)))
        h1 = matrix(np.concatenate((h1_31, h1_32, h1_33)))
        'Solving QPP using cvxopt package and module solvers'
        sol_1 = solvers.qp(P1,q1,G1,h1)
        alpha_gamma = sol_1['x'][0:n_label_neg]
        #solution
        z_1 = -np.dot(np.dot(d1,F.T),alpha_gamma)
```

Next, continue the process to obtain the solution for the negative hyperplane.

```
        #consider the problem
        ##using H, F and J from above
        #construct (H.TH+c_2I+c_3J.TLJ)^-1
        s2 = matrix(np.dot(F.T,F) + self.c_2*(np.identity(np.size(F
            ,1)))+self.c_3*np.dot(np.dot(J.T,L),J))
        d2 = matrix(np.linalg.inv(s2))
        ' Set the problem into the form of cvxopt '
        #construct P
        P2_11 = matrix(np.dot(np.dot(H,d2),H.T))
        P2_12 = matrix(np.zeros((n_label_pos,n_label_pos)))
        P2_21 = matrix(np.zeros((n_label_pos,n_label_pos)))
        P2_22 = matrix(np.zeros((n_label_pos ,n_label_pos)))
        P2_r1 = matrix(np.concatenate((P2_11, P2_12), axis=1))
        P2_r2 = matrix(np.concatenate((P2_21, P2_22), axis=1))
        P2 = matrix(np.concatenate((P2_r1, P2_r2)))
        # construct q
        q2_11 = (-1)*matrix(np.ones((n_label_pos,1)))
        q2_12 = matrix(np.zeros((n_label_pos,1)))
        q2 = matrix(np.concatenate((q2_11, q2_12)))
        #construct G
```

```
        G2_11 = (-1/self.tau_1)*matrix(np.identity(n_label_pos))
        G2_12 = (1+1/self.tau_1)*matrix(np.identity(n_label_pos))
        G2_21 = matrix(np.zeros((n_label_pos,n_label_pos )))
        G2_22 = (-1)*matrix(np.identity(n_label_pos))
        G2_31 = matrix(np.identity(n_label_pos))
        G2_32 = (-1)*matrix(np.identity(n_label_pos))

        G2_r1 = matrix(np.concatenate((G2_11, G2_12), axis=1))
        G2_r2 = matrix(np.concatenate((G2_21, G2_22), axis=1))
        G2_r3 = matrix(np.concatenate((G2_31, G2_32), axis=1))
        G2 = matrix(np.concatenate((G2_r1, G2_r2, G2_r3)))
        #construct h
        h2_31 = self.c_1*matrix(np.ones((n_label_pos,1)))
        h2_32 = matrix(np.zeros((n_label_pos,1)))
        h2_33 = matrix(np.zeros((n_label_pos ,1)))
        h2 = matrix(np.concatenate((h2_31, h2_32, h2_33)))
        'Solving QPP using cvxopt package and module solvers'
        sol_2 = solvers.qp(P2,q2,G2,h2)
        beta_pho = sol_2['x'][0:n_label_pos]
        #solution
        z_2 = np.dot(np.dot(d2,H.T),beta_pho)
```

After, we get the solutions of the problems, then $w_1, b_1$ and $w_2, b_2$ are obtain by the following code.

```
        #the optimal solution of positive hyperplane
        w_1 = z_1[0:-1]
        b_1 = z_1[-1]
        #the optimal solution of negative hyperplane
        w_2 = z_2[0:-1]
        b_2 = z_2[-1]
        self.w_1 = w_1
        self.b_1 = b_1
        self.w_2 = w_2
        self.b_2 = b_2
        return self
```

Here is the complete code for constructing and using QPP to solve our problems. Then, we will see the criteria to assign the new data, along with its corresponding class.

Once the optimal hyperplanes $\mathbf{x}^\top \mathbf{w}_1 + b_1 = 0$ and $\mathbf{x}^\top \mathbf{w}_2 + b_2 = 0$ are obtained form (3.1.18) and (3.1.19), a new sample point $\mathbf{x} \in \mathbb{R}^n$ is assigned to the positive class or negative class, depending on which of the two hyperplanes is closer to $x$, i.e.

$$\text{class}(i) = \arg\min_{i=1,2} \frac{|\mathbf{x}^\top \mathbf{w}_i + b_i|}{\|\mathbf{w}_i\|}. \tag{3.1.20}$$

Here, $|.|$ denote the perpendicular distance of the data from the hyperplane. The pseudocode for a linear case is shown in Algorithm 1.

**Algorithm 1** Laplacian twin support vector machine with pinball loss function

**Input:** Matrix Data $X$,
Labeled vector $y$,
model parameters $c_1, c_2, c_3, \tau, k$.

1. Generate data $X$ into positive labeled $A$, negative labeled $B$ and unlabeled data.
2. Obtain the adjacency matrix and then the graph Laplacian matrix $L$.
3. Find the solution $\omega^*$ and $\mu^*$ of the problem (3.1.16) and (3.1.17) by solving QPPs.
4. Compute $\mathbf{w}_1, \mathbf{w}_2, b_1, b_2$ by calculate the $z_1$, $z_2$ from (3.1.18) and (3.1.19).
5. Obtain two optimal hyperplane $\mathbf{x}^\top \mathbf{w}_1 + b_1 = 0$ and $\mathbf{x}^\top \mathbf{w}_2 + b_2 = 0$.

**Output:** For a new sample $\mathbf{x}$, it is assigned to be the positive or negative class by considering the eq. (3.2.18)

### 3.1.2 Nonlinear Case

We are using a kernel trick [35, 36] to expand our linear Lap-PTSVM to the nonlinear case. The kernel trick is used to map the input data points into a higher-dimensional feature space. If $K(\cdot, \cdot)$ is the predefined kernel function, then the nonparallel hyperplanes in the kernel-generated space can be written as follows:

$$K(\mathbf{x}^\top, M^\top)\mathbf{w}_1 + b_1 = 0 \text{ and } K(\mathbf{x}^\top, M^\top)\mathbf{w}_2 + b_2 = 0,$$

where $\mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^{l+u}$. The two nonparallel hyperplanes above are produced by the following QPPs:

$$\min_{\mathbf{w}_1, b_1, \boldsymbol{\xi}} \frac{1}{2}\|K(A, M^\top)\mathbf{w}_1 + \mathbf{e}_1 b_1\|^2 + c_1 \mathbf{e}_2^\top \boldsymbol{\xi} + \frac{c_2}{2}(\mathbf{w}_1^\top K(M, M^\top)\mathbf{w}_1 + b_1^2)$$
$$+ \frac{c_3}{2}(\mathbf{w}_1^\top K(M, M^\top) + \mathbf{e}^\top b_1) L (K(M, M^\top)\mathbf{w}_1 + \mathbf{e} b_1)$$

$$\begin{aligned} \text{s.t.} \quad & -(K(B,M^\top)\mathbf{w}_1+\mathbf{e}_2 b_1) \geq \mathbf{e}_2-\boldsymbol{\xi}, \\ & -(K(B,M^\top)\mathbf{w}_1+\mathbf{e}_2 b_1) \leq \mathbf{e}_2+\frac{\boldsymbol{\xi}}{\tau}, \\ & \boldsymbol{\xi} \geq \mathbf{0}, \end{aligned} \tag{3.1.21}$$

and

$$\begin{aligned} \min_{\mathbf{w}_2,b_2,\boldsymbol{\xi}} \quad & \frac{1}{2}\|K(B,M^\top)\mathbf{w}_2+\mathbf{e}_2 b_2\|^2+c_1\mathbf{e}_1^\top\boldsymbol{\xi}+\frac{c_2}{2}(\mathbf{w}_2^\top K(M,M^\top)\mathbf{w}_2+b_2^2) \\ & +\frac{c_3}{2}(\mathbf{w}_2^\top K(M,M^\top)+\mathbf{e}^\top b_2)L(K(M,M^\top)\mathbf{w}_2+\mathbf{e}b_2) \\ \text{s.t.} \quad & K(A,M^\top)\mathbf{w}_2+\mathbf{e}_1 b_2 \geq \mathbf{e}_1-\boldsymbol{\xi}, \\ & K(A,M^\top)\mathbf{w}_2+\mathbf{e}_1 b_2 \leq \mathbf{e}_1+\frac{\boldsymbol{\xi}}{\tau}, \\ & \boldsymbol{\xi} \geq \mathbf{0}. \end{aligned} \tag{3.1.22}$$

Their dual problems can be obtained as

$$\begin{aligned} \min_{\boldsymbol{\omega}} \quad & \frac{1}{2}\boldsymbol{\omega}^\top F_\Phi(H_\Phi^\top H_\Phi+c_2O_\Phi+c_3J_\Phi^\top LJ_\Phi)^{-1}F_\Phi^\top\boldsymbol{\omega})-\boldsymbol{\omega}^\top\mathbf{e}_2 \\ \text{s.t.} \quad & \boldsymbol{\omega}+\boldsymbol{\beta}\left(1+\frac{1}{\tau}\right) \leq c_1\mathbf{e}_2, \\ & \boldsymbol{\omega}+\boldsymbol{\beta} \geq 0,\ \boldsymbol{\beta} \geq 0, \end{aligned} \tag{3.1.23}$$

and

$$\begin{aligned} \min_{\boldsymbol{\mu}} \quad & \frac{1}{2}\boldsymbol{\mu}^\top H_\Phi(F_\Phi^\top F_\Phi+c_2O_\Phi+c_3J_\Phi^\top LJ_\Phi)^{-1}H_\Phi^\top\boldsymbol{\mu}-\boldsymbol{\mu}^\top\mathbf{e}_1 \\ \text{s.t.} \quad & \boldsymbol{\mu}+\boldsymbol{\rho}\left(1+\frac{1}{\tau}\right) \leq c_1\mathbf{e}_1, \\ & \boldsymbol{\mu}+\boldsymbol{\rho} \geq 0,\ \boldsymbol{\rho} \geq 0, \end{aligned} \tag{3.1.24}$$

where $\boldsymbol{\omega}=\boldsymbol{\alpha}-\boldsymbol{\beta}$, $\boldsymbol{\mu}=\boldsymbol{\eta}-\boldsymbol{\rho}$ and $\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\eta}$ and $\boldsymbol{\rho}$ are Lagrange multipliers and $H_\Phi=\begin{bmatrix}K(A,M^\top) & \mathbf{e}_1\end{bmatrix}$, $O_\Phi=\begin{bmatrix}K(M,M^\top) & 0\\ 0 & 1\end{bmatrix}$, $F_\Phi=\begin{bmatrix}K(B,M^\top) & \mathbf{e}_2\end{bmatrix}$, $J_\Phi=\begin{bmatrix}K(M,M^\top) & \mathbf{e}\end{bmatrix}$. The two hyperplanes are obtain from the solution $\boldsymbol{\omega}^*$ and $\boldsymbol{\mu}^*$ of (3.1.23) and (3.1.24) by

$$K(\mathbf{x}^\top,M^\top)\mathbf{w}_1+b_1=0,$$

where

$$\begin{bmatrix}\mathbf{w}_1\\ b_1\end{bmatrix}=-(H_\Phi^\top H_\Phi+c_2O_\Phi+c_3J_\Phi^\top LJ_\Phi)^{-1}F_\Phi^\top\boldsymbol{\omega}^*, \tag{3.1.25}$$

and

$$K(\mathbf{x}^\top, M^\top)\mathbf{w}_2 + b_2 = 0,$$

where

$$\begin{bmatrix} \mathbf{w}_2 \\ b_2 \end{bmatrix} = (F_\Phi^\top F_\Phi + c_2 O_\Phi + c_3 J_\Phi^\top L J_\Phi)^{-1} H_\Phi^\top \boldsymbol{\mu}^*. \tag{3.1.26}$$

After obtaining the pair of nonparallel hyperplanes, a new sample point $\mathbf{x} \in \mathbb{R}^n$ is assigned to the positive or negative class in a manner similar to the linear situation. The pseudocode for a non-linear case is similar to the linear case.

### 3.1.3 Noise Insensitivity

Now, we discuss how Lap-PTSVM solves the noise sensitivity issue. For shortness, we consider the linear case and deal with the problem (3.1.1). However, the same analysis also applies to the nonlinear case. Define the generalized sign function $\mathrm{sgn}_\tau(v)$ as

$$\mathrm{sgn}_\tau(v) = \begin{cases} 1 & \text{if } v > 0, \\ [-\tau, 1] & \text{if } v = 0, \\ -\tau & \text{if } v < 0, \end{cases} \tag{3.1.27}$$

where $v = (1 - y(\mathbf{w}^\top \mathbf{x} + b))$ and $\mathrm{sgn}_\tau(1 - y(\mathbf{w}^\top \mathbf{x} + b))$ is the subgradient of the pinball loss function. The optimality condition for equation (3.1.1) can be expressed as:

$$\begin{aligned} \mathbf{0} \in & A^\top (A\mathbf{w}_1 + \mathbf{e}_1 b_1) + c_2 \mathbf{w}_1 + c_3 M^\top L (M\mathbf{w}_1 + \mathbf{e} b_1) \\ & + c_1 \sum_{i=1}^{m_2} \mathrm{sgn}_\tau(1 + (\mathbf{w}_1 \mathbf{x}_i^- + b_1)) \mathbf{x}_i^-, \end{aligned} \tag{3.1.28}$$

where $\mathbf{0}$ is zero vector and $\mathbf{x}_i^- \in B$. For given $\mathbf{w}_1$ and $b_1$, we can be separate the index set of $B$ into three sets:

$$E_1^+ = \{i : 1 + (\mathbf{w}_1^\top \mathbf{x}_i^- + b_1) > 0\},$$
$$E_2^+ = \{i : 1 + (\mathbf{w}_1^\top \mathbf{x}_i^- + b_1) < 0\},$$

$$E_3^+ = \{i : 1 + (\mathbf{w}_1^\top \mathbf{x}_i^- + b_1) = 0\}.$$

Here, $i \in \{1, 2, 3, \ldots, m_2\}$.

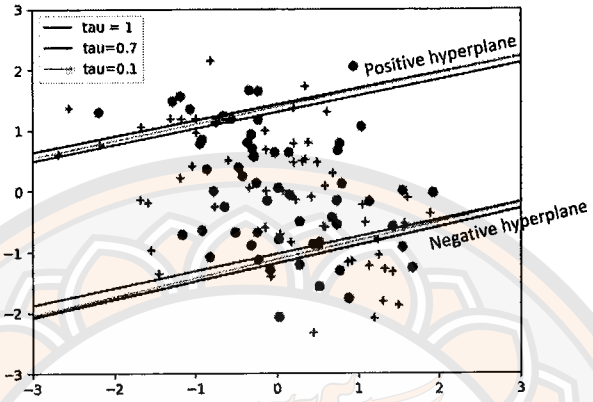



**Figure 16 Illustration of Lap-PTSVM generating nonparallel hyperplanes using different of $\tau$.**

Using the notation $E_1^+, E_2^+$ and $E_3^+$, the equation (3.2.20) can be rewritten as the existence of $\theta_i \in [-\tau, 1]$, such that

$$\frac{1}{c_1} A^\top (A\mathbf{w}_1 + \mathbf{e}_1 b_1) + \frac{c_2}{c_1}\mathbf{w}_1 + \frac{c_3}{c_1} M^\top L(M\mathbf{w}_1 + \mathbf{e}b_1) - \tau \sum_{i \in E_2^+} \mathbf{x}_i^- + \sum_{i \in E_1^+} \mathbf{x}_i^- + \theta_i \sum_{i \in E_3^+} \mathbf{x}_i^- = 0.$$

The above notation indicates that $\tau$ controls the number of points in $E_1^+, E_2^+$ and $E_3^+$. When $\tau$ is small, there are a lot of points in $E_2^+$, therefore the number of points in $E_1^+$ and $E_3^+$ will be small as shown in figure 16 and 17. As a result, feature noise around the decision boundary will have a significant impact on the classification outcome. When $\tau$ becomes large, all three sets have a large number of points, resulting in a lower sensitivity.

**Proposition 3.1.1.** If the optimization problem (3.1.3) or (3.1.21) has a solution then the following inequalities must hold:

$$-\frac{\mathbf{e}_1^\top (A\mathbf{w}_1 + \mathbf{e}_1 b_1) + c_2 b_1 + c_3 \mathbf{e}^\top L(M\mathbf{w}_1 + \mathbf{e}b_1)}{c_1 m_2} \leq 1,$$

and

$$\frac{p_1}{m_2} \leq 1 - \frac{1 + \frac{\mathbf{e}_1^\top (A\mathbf{w}_1 + \mathbf{e}_1 b_1) + c_2 b_1 + c_3 \mathbf{e}^\top L(M\mathbf{w}_1 + \mathbf{e}b_1)}{c_1 m_2}}{1+\tau},$$

where $p_1$ denote the number samples in $E_1^+$.

*Proof.* Let $\mathbf{x}_{i_0}^- \in E_1^+$ be an arbitrary negative point. From the KKT condition (3.1.10) and (3.1.11), we have $\beta_{i_0} = \gamma_{i_0} = 0$. From the KKT condition (3.1.8), we can obtain $\alpha_{i_0} = c_1$, then $\alpha_{i_0} - \gamma_{i_0} = c_1$. Let $\lambda = \boldsymbol{\alpha} - \boldsymbol{\gamma}$ and, subsequently, $\lambda_{i_0} = c_1$. Also, we get $-\big(\mathbf{e}_1^\top (A\mathbf{w}_1 + \mathbf{e}_1 b_1) + c_2 b_1 + c_3 \mathbf{e}^\top L(M\mathbf{w}_1 + \mathbf{e}b_1)\big) = p_1 c_1 + \sum_{i \notin E_1^+} \lambda_i$ from (3.1.7).
Since $\boldsymbol{\alpha}_i \geq 0$ and $\gamma_i \geq 0$, we can have $-c_1\tau \leq \lambda_i \leq c_1$. Therefore, we get

$$p_1 c_1 \leq -\big(\mathbf{e}_1^\top (A\mathbf{w}_1 + \mathbf{e}_1 b_1) + c_2 b_1 + c_3 \mathbf{e}^\top L(M\mathbf{w}_1 + \mathbf{e}b_1)\big) + c_1\tau(m_2 - p_1),$$

thus,

$$-\frac{\mathbf{e}_1^\top (A\mathbf{w}_1 + \mathbf{e}_1 b_1) + c_2 b_1 + c_3 \mathbf{e}^\top L(M\mathbf{w}_1 + \mathbf{e}b_1)}{c_1 m_2} \leq 1$$

and

$$1 + \tau \leq -\frac{\big(\mathbf{e}_1^\top (A\mathbf{w}_1 + \mathbf{e}_1 b_1) + c_2 b_1 + c_3 \mathbf{e}^\top L(M\mathbf{w}_1 + \mathbf{e}b_1)\big)}{c_1 p_1} + \frac{c_1 \tau c_1 m_2}{c_1 p_1}.$$

Finally, we have

$$\frac{p_1}{m_2} \leq 1 - \frac{1 + \frac{\mathbf{e}_1^\top (A\mathbf{w}_1 + \mathbf{e}_1 b_1) + c_2 b_1 + c_3 \mathbf{e}^\top L(M\mathbf{w}_1 + \mathbf{e}b_1)}{c_1 m_2}}{1+\tau},$$

therefore proving our argument. □

As a result, the upper bound on the number of samples in $E_1^+$ is given by the preceding proposition, which also takes into consideration the Laplacian term $c_3 \mathbf{e}^\top L(M\mathbf{w}_1 + \mathbf{e}b_1)$. Since a lot fewer data samples are distributed in sets other than $E_2^+$, $p_1$decreases as $\tau$ drops, the result becomes more sensitive to feature noise. This conclusion can be observed in Figure 17.

From the above, a similar analysis would go for the other problems of Lap-PTSVM. Therefore, we also have a proposition for the second problem (3.1.2) as

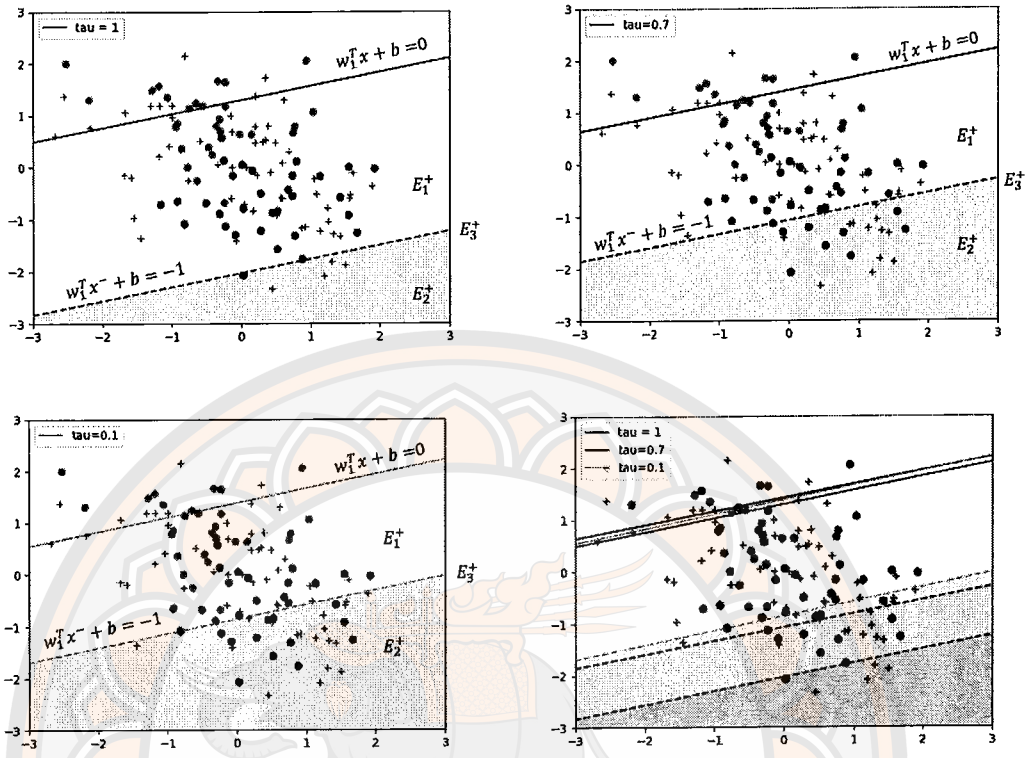



**Figure 17 Illustration the set $E_1^+, E_2^+$ and $E_3^+$ when using different of $\tau$.**

follows. The index set for set $A$ can be partitioned into three sets for given $w_2, b_2$ and denote that $x_i^+ \in A$:

$$E_1^- = \{i : 1 - (\mathbf{w}_2^\top \mathbf{x}_i^+ + b_2) < 0\},$$
$$E_2^- = \{i : 1 - (\mathbf{w}_2^\top \mathbf{x}_i^+ + b_2) > 0\},$$
$$E_3^- = \{i : 1 - (\mathbf{w}_2^\top \mathbf{x}_i^+ + b_2) = 0\},$$

where $i = 1, \ldots, m_1$. As a result, we get the following proposition.

**Proposition 3.1.2.** If the optimization problem (3.1.4) or (3.1.22) has a solution then the following inequalities must hold:

$$-\frac{\mathbf{e}_2^\top (B\mathbf{w}_2 + \mathbf{e}_2 b_2) + c_2 b_2 + c_3 \mathbf{e}^\top L(M\mathbf{w}_2 + \mathbf{e}b_2)}{c_1 m_1} \leq 1,$$

and

$$\frac{p_2}{m_1} \leq 1 - \frac{1 - \frac{\mathbf{e}_2^\top (B\mathbf{w}_2 + \mathbf{e}_2 b_2) + c_2 b_2 + c_3 \mathbf{e}^\top L(M\mathbf{w}_2 + \mathbf{e}b_2)}{c_1 m_1}}{1 + \tau},$$

where $p_2$ denote the number of positive samples $\mathbf{x}_i^+$ in $E_1^-$.

From Figure 18, we can observe that the region between the two hyperplanes of Lap-TSVM and Lap-PTSVM is quite large; this may be due to the influence of the Laplacian term on the construction of the hyperplanes. Although Lap-TSVM in Figure 18 can accurately classify the data points, the two hyperplanes of the model are not suitable, as it is far away from two class samples. However, in the proposed Lap-PTSVM, the region between the two hyperplanes is investigated by controlling parameter $\tau$. As $\tau$ decrease, there is a few label point in that area. When $\tau$ increases, there are more label points, as illustrated in Figure 18. As discussed in the above proposition, the number of label points has an effect on constructing the hyperplanes. Consequently, Lap-PTSVM has the ability to control the number of points in the area of the two hyperplanes by adjusting parameter $\tau$, which decreases the noise sensitivity of the model. Moreover, our Lap-PTSVM provides the appropriate hyperplanes.

### 3.1.4 Scatter minimization

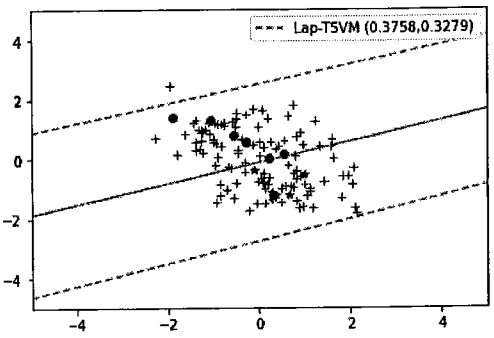
Scatter minimization can also be used to understand the Lap-PTSVM. For simplicity, consider only the first QPP (3.1.1) of the Lap-PTSVM. The conclusions for another QPP (3.1.2) can be gotten so also. For given $\mathbf{x}_i^- \in B$ and $\mathbf{x}_j^+ \in A$, the positive hyperplane $\mathbf{w}_1^\top \mathbf{x} + b_1 = 0$ be established by data samples under subset $Y_2^+ \subseteq A$. The two hyperplanes $\mathcal{H}^+ = \{\mathbf{w}_1^\top \mathbf{x}_i^- + b_1 + 1 = 0\}$ and $\mathcal{H}^- = \{\mathbf{w}_2^\top \mathbf{x}_j^+ + b_2 - 1 = 0\}$ are defined by data samples in subsets $Y_3^+ \subseteq E_3^+$ and subset $Y_3^- \subseteq E_3^-$, respectively.

The scatter is calculated by adding the distances between each point $\mathbf{x}_i^-$ and one supplied negative sample $\mathbf{x}_{i_3}^- \in Y_3^+$. The scatter of $\mathbf{x}_i^- \in B$ around the sample $\mathbf{x}_{i_3}^-$ can be determined as

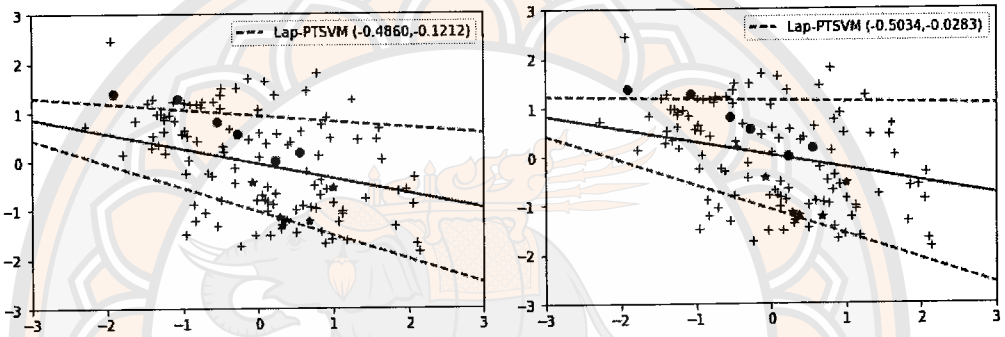
$$\sum_{i=1}^{m_2} |\mathbf{w}_1^\top \mathbf{x}_{i_3}^- + b_1 - (\mathbf{w}_1^\top \mathbf{x}_i^- + b_1)| = \sum_{i=1}^{m_2} |\mathbf{w}_1^\top (\mathbf{x}_{i_3}^- - \mathbf{x}_i^-)|. \tag{3.1.29}$$

We get the following equation by using $\mathbf{w}_1^\top \mathbf{x}_{i_3}^- + b_1 + 1 = 0$:

$$\sum_{i=1}^{m_2} |\mathbf{w}_1^\top (\mathbf{x}_{i_3}^- - \mathbf{x}_i^-)| \quad = \sum_{i=1}^{m_2} |\mathbf{w}_1^\top \mathbf{x}_{i_3}^- + b_1 - (\mathbf{w}_1^\top \mathbf{x}_i^- + b_1)|$$




(a) Lap-TSVM




(b) Lap-PTSVM ($\tau = 0.1$) (c) Lap-PTSVM ($\tau = 0.8$)

**Figure 18 Illustrations of classification results on a 2D artificial data set. The two optimal hyperpalnes and the middle line between them are represented by dash line and solid line, respectively.**

$$= \sum_{i=1}^{m_2} |-1-(\mathbf{w}_1^\top \mathbf{x}_i^- + b_1)|$$
$$= \sum_{i=1}^{m_2} |1+(\mathbf{w}_1^\top \mathbf{x}_i^- + b_1)|.$$

Similarly, using a specific data sample $\mathbf{x}_{j_2}^+ \in Y_2^+$, the scatter for every sample $\mathbf{x}_j^+ \in A$ is calculated as follows:

$$\sum_{j=1}^{m_1} |\mathbf{w}_1^\top (\mathbf{x}_{j_2}^+ - \mathbf{x}_j^+)| = \sum_{j=1}^{m_1} |\mathbf{w}_1^\top \mathbf{x}_{j_2}^+ + b_1 - (\mathbf{w}_1^\top \mathbf{x}_j^+ + b_1)|$$
$$= \sum_{i=1}^{m_1} |-(\mathbf{w}_1^\top \mathbf{x}_j^+ + b_1)|,$$

where $\mathbf{w}_1^\top \mathbf{x}_{j_2}^+ + b_1 = 0$. Because the scatter is a positive value, we can take it as the sum of squares, i.e., $\sum_{i=1}^{m_1}(-(\mathbf{w}_1^\top \mathbf{x}_j^+ + b_1))^2$.

Consider the formula as follows:

$$\min_{\mathbf{w}_1, b_1} \frac{1}{2}\sum_{i=1}^{m_1}(-(\mathbf{w}_1^\top \mathbf{x}_j^+ + b_1))^2 + \frac{c_2}{2}(\|\mathbf{w}_1\|^2 + b_1^2) + \frac{c_3}{2}(M\mathbf{w}_1 + \mathbf{e}b_1)^\top L(M\mathbf{w}_1 + \mathbf{e}b_1) + c_{11}\sum_{i=1}^{m_2}|1 + (\mathbf{w}_1^\top \mathbf{x}_i^- + b_1)|, \tag{3.1.30}$$

where $c_{11}$ is a constant. We confirm that the scatters $\mathbf{x}_j^+ \in A$ about the hyperplane $\mathbf{w}_1^\top \mathbf{x} + b_1 = 0$ are minimized due to the expression of the first term. However, equation (3.1.30) implements structural risk minimization in the second term and incorporates the effects of Laplacian regularization in the third term. Meanwhile, the last term aims to minimize the error values caused by the proximity of $B$ samples to $\mathcal{H}^+$ by reducing the scatter of $\mathbf{x}_i^- \in B$ around the hyperplane $\mathcal{H}^+$.

The first three terms of Eq. (3.1.30) are expressed within the Lap-PTSVM framework (Eq. (3.1.1)) in their mathematically equivalent form. Additionally, the absolute value employed in Eq. (3.1.30) is extended to $L_\tau$. Specifically, this involves introducing the misclassification term $c_{12}\mathcal{L}_{hinge}(1 + (\mathbf{w}_1^\top \mathbf{x}_i^- + b_1))$ into (3.1.30), we obtain the Lap-PTSVM with $c_1 = c_{11} + c_{12}$ and $\tau = \frac{c_{11}}{c_1}$. Thus the proper range of $\tau$ is $0 \leq \tau \leq 1$, according to interpretation.

Small within-class scatter and small misclassification error are two desirable aims for a good classifier. The hinge loss emphasizes within-class scatter, while the pinball loss is a compromise between the two aims. It can be analyzed in the same way for $\mathbf{x}_j^- \in \mathcal{H}_-$.

### 3.1.5 Numerical Experiments

For the experiments, we test a performance of Lap-PTSVM on artificial datasets, 17 benchmark datasets, and compare the efficiency of our model with TSVM, Pin-TSVM and Lap-TSVM. Both linear and nonlinear cases are explored. For nonlinear case, we use the RBF kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\big(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\big)$, where $\sigma$ is a parameter. All experiments are implemented in Python 3.9.5. on Windows 8 running on a 1.9 GHz laptop with 4 GB RAM with system configuration Intel

Core i5+ Duo CPU E7500 (2.93 GHz). Non-parametric statistical test are also conducted to justify the competitive performance of Lap-PTSVM.

**Table 1 Description of UCI datasets.**

| Datasets | No.of Feature | No. of Sample | Imbalance ratio |
|---|---|---|---|
| Appendicitis | 7 | 106 | 4.05 |
| Australian | 14 | 690 | 1.25 |
| Breast | 10 | 116 | 1.23 |
| Bupa | 6 | 345 | 1.38 |
| Banknote | 5 | 1372 | 1.25 |
| Diabetes | 8 | 768 | 1.87 |
| Fertility | 9 | 100 | 7.33 |
| Heart-Statlog | 13 | 270 | 1.25 |
| Heart-C | 13 | 303 | 1.19 |
| Ionosphere | 33 | 351 | 1.79 |
| Monk-2 | 6 | 432 | 1.12 |
| Monk-3 | 6 | 432 | 1.12 |
| WDBC | 30 | 569 | 1.68 |
| Pima | 8 | 768 | 1.86 |
| Spectheart | 22 | 267 | 3.85 |
| Sonar | 60 | 208 | 1.14 |
| Titanic | 3 | 2201 | 2.09 |

### 3.1.6 Artificial datasets

We conduct the tests on a two-dimensional in which samples are drawn with equal probability using two Gaussian distributions: $\mathbf{x}_i, i \in \{i : y_i = 1\} \sim \mathcal{N}(\mu_1, \Sigma_1)$ and $\mathbf{x}_i, i \in \{i : y_i = -1\} \sim \mathcal{N}(\mu_2, \Sigma_2)$ with $\mu_1 = [0.5, -3]^\top, \mu_2 = [-0.5, 3]^\top$ and $\Sigma_1 = \Sigma_2 = [0.2\ 3; 0\ 3]$. The noise points' labels are chosen with uniform probability from the range $\{1, -1\}$. This noise sample was obtained from a Gaussian distribution $\mathcal{N}(\mu_n, \Sigma_n)$, with $\mu_n = [0, 0]^\top$ and $\Sigma_n = [1\ -0.8; -0.8\ 1]$. The ratio of noise data , represented by $r$, control the level of noise to total samples in the original distribution.

The noise samples have an effect on the labels at the boundary as we can see in the results mentioned in Figure 19. It depicts the classification results produced by Lap-TSVM and our Lap-PTSVM with various noises. Here, we use 10% of labeled data, red circle and green star represent labeled data from class $+1$ and $-1$, respectively. For the red and green plus are represented unlabeled data. As we arise the amount of noise from $r = 0$ to $r = 0.1$, the two hyperplanes of Lap-TSVM diverge from $0.6903, 0.5033$ to $0.1292, 0.2232$, respectively, whereas in Lap-PTSVM, it diverges from $0.6064, 0.2038$ to $0.2614, 0.004$, respectively. We found that it has slightly changed in our Lap-PTSVM. Additionally, we evaluate the absolute value of slope changing of each model as noise increases in Table 2. $\mathbf{w}_i^r$ denoted a slope of the $i$-hyperplane corresponding to noise $r$. The smaller absolute value is in bold. As in the table, the slope change in Lap-PTSVM yields less deviation than Lap-TSVM in all cases. As it evident, this implies that the proposed model is significantly insensitive to noise near the decision boundary.

**Table 2 Absolute values of slope changing in each model.**

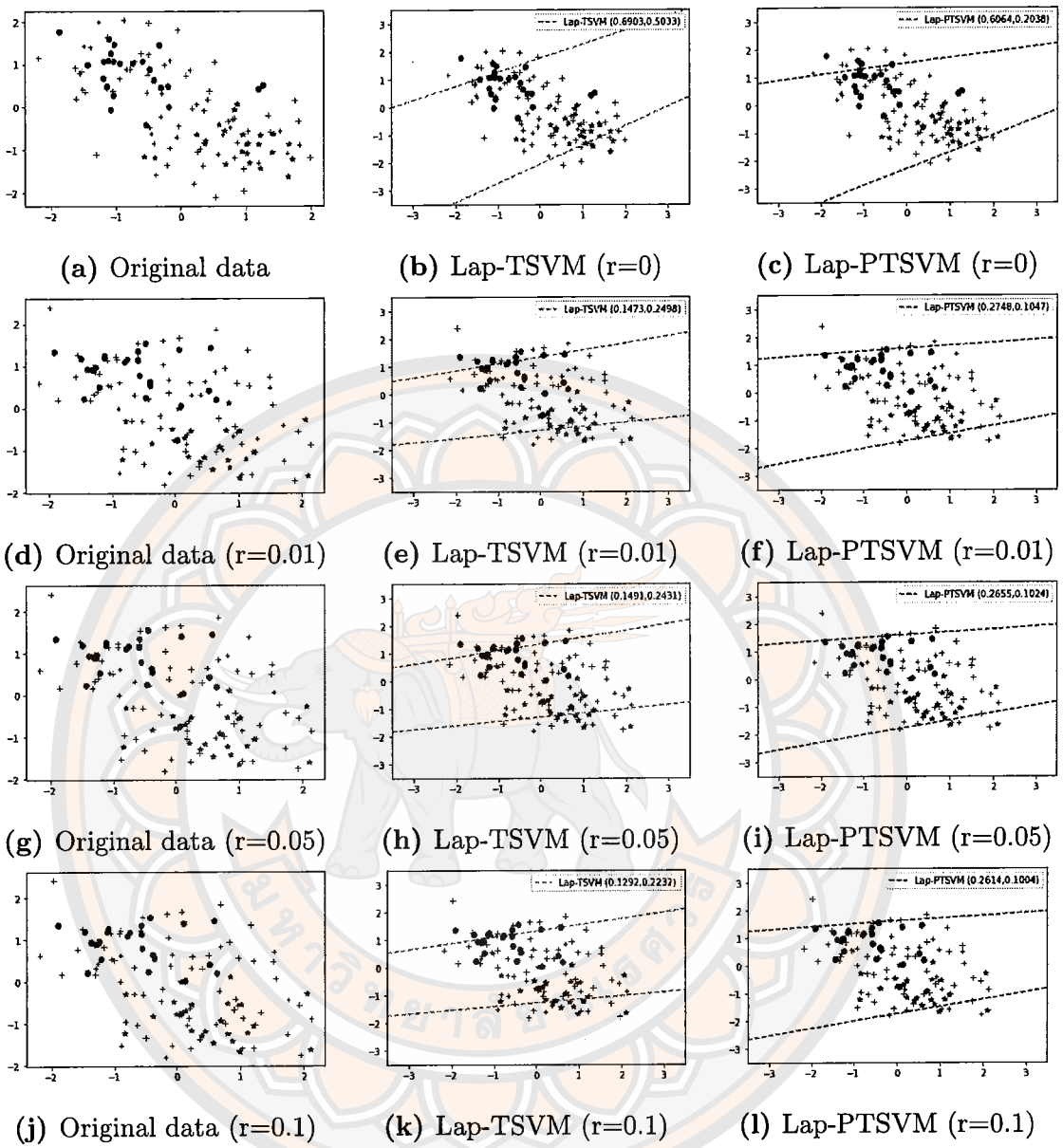
| Absolute value | i=1 | | i=2 | |
|---|---|---|---|---|
| | Lap-TSVM | Lap-PTSVM | Lap-TSVM | Lap-PTSVM |
| $\lvert w_i^0 - w_i^{0.05}\rvert$ | 0.543 | **0.3316** | 0.2535 | **0.0991** |
| $\lvert w_i^{0.05} - w_i^{0.1}\rvert$ | 0.0437 | **0.0093** | 0.0104 | **0.0023** |
| $\lvert w_i^{0.1} - w_i^{0.2}\rvert$ | 0.0199 | **0.0041** | 0.0199 | **0.002** |

### 3.1.7 UCI datasets

The experiments on 17 binary classification datasets are shown to illustate efficiency of our proposed work. Table 1 list the data descriptions and also provides the imbalance ratio of each datasets, which can be calculated from the following formula:

$$\text{Imbalance ratio} = \frac{\text{No. of sample in the majority class}}{\text{No. of sample in the minority class}}$$

Before training, we normalized the features of the data samples in the interval




(a) Original data (b) Lap-TSVM (r=0) (c) Lap-PTSVM (r=0)

(d) Original data (r=0.01) (e) Lap-TSVM (r=0.01) (f) Lap-PTSVM (r=0.01)

(g) Original data (r=0.05) (h) Lap-TSVM (r=0.05) (i) Lap-PTSVM (r=0.05)

(j) Original data (r=0.1) (k) Lap-TSVM (r=0.1) (l) Lap-PTSVM (r=0.1)

**Figure 19 The classification results produced by Lap-TSVM and our proposed Lap-PTSVM on an artificial dataset with level of noise.**

$[0, 1]$ and used the grid search technique [38] to optimize the trade-off parameters and kernel parameter $\sigma$. We have chosen parameter values $c_1, c_2$ and $c_3$ from the set $\{2^i | i = -4, -3, -2, -1, 0, 1, 2, 3, 4\}$ and $\tau$ from the set $\{0.1, 0.2, 0.3, 0.5, 0.7, 1\}$. The parameter of RBF kernel $\sigma$ is found over the range $\{10^i | i = -2, -1, 0, 1, 2\}$. The classification results obtained from 10-fold cross validation strategy.

**Table 3 The mean and standard deviation of tests accuracy with various labeled ratios on the UCI dataset were calculated using a linear kernel.**

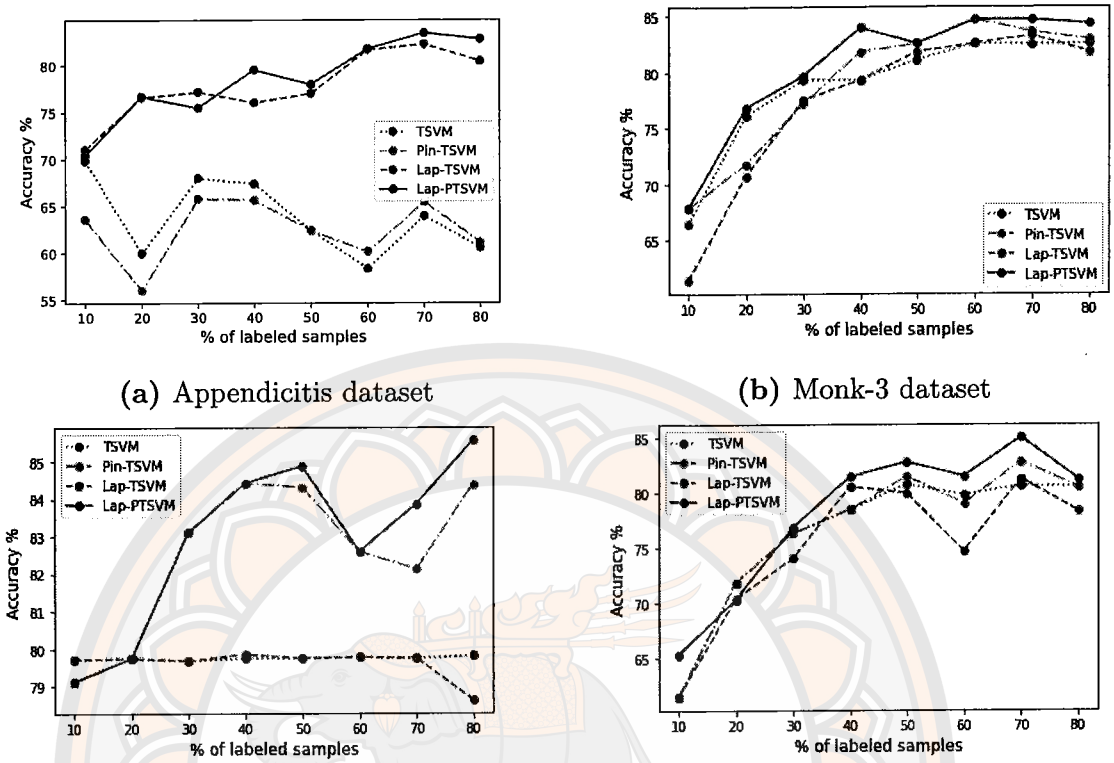
| Datasets | Ratio (%) | Existing model | | | Proposed model |
|---|---|---|---|---|---|
| | | TSVM | Pin-TSVM | Lap-TSVM | Lap-PTSVM |
| Appendicitis | 30 | 80.95±13.30 | 80.95±13.30 | 80.95±13.29 | **82.47±4.81** |
| | 50 | 80.95±14.25 | 80.95±14.25 | **83.81±11.85** | 82.38±12.38 |
| | 70 | 82.61±12.42 | 82.62±12.42 | 82.63±10.65 | **84.05±11.86** |
| Australian | 30 | 86.58±2.79 | 86.13±3.14 | 86.80±2.88 | **87.02±2.77** |
| | 50 | **86.56±4.31** | **86.56±4.31** | **86.56±4.31** | **86.56±4.31** |
| | 70 | 87.00±4.64 | **87.23±4.31** | 86.32±4.82 | 87.00±4.56 |
| Breast | 30 | 55.18±18.67 | 55.18±18.67 | **56.77±19.64** | 53.57±18.95 |
| | 50 | 55.54±13.37 | 55.54±13.37 | 63.57±16.12 | **67.86±11.49** |
| | 70 | 55.54±15.54 | 55.54±15.54 | 59.64±16.62 | **64.82±16.68** |
| Bupa | 30 | 57.55±6.83 | 61.56±5.98 | 57.57±9.56 | **61.92±10.12** |
| | 50 | 60.71±9.22 | 62.02±6.94 | 57.51±13.39 | **66.96±11.91** |
| | 70 | 47.21±9.37 | 62.89±8.70 | 57.73±9.17 | **63.83±9.91** |
| Heart-Statlog | 30 | 68.04±8.28 | 65.82±9.06 | **77.22±9.55** | 75.53±9.27 |
| | 50 | 62.42±10.32 | 62.38±7.51 | 77.06±10.83 | **78.24±10.95** |
| | 70 | 63.95±7.75 | 59.47±8.19 | 82.29±7.86 | **83.50±8.11** |
| Ionosphere | 30 | 80.65±5.52 | 81.09±6.73 | **85.93±4.24** | 81.05±6.00 |
| | 50 | 78.42±6.01 | 79.76±7.06 | **81.52±8.98** | 79.31±5.49 |
| | 70 | 80.16±6.09 | 76.17±5.20 | **86.67±7.09** | 81.07±8.38 |
| Monk-2 | 30 | **82.50±10.03** | 81.43±9.95 | 80.36±8.34 | 82.14±9.03 |
| | 50 | 81.43±6.14 | 83.21±3.59 | 83.93±6.44 | **85.00±4.46** |
| | 70 | 81.43±7.79 | 83.21±6.59 | 82.50±5.63 | **83.21±7.15** |
| Monk-3 | 30 | 79.29±6.14 | 77.14±5.80 | 77.50±8.15 | **79.64±6.97** |
| | 50 | 81.07±10.47 | 82.50±9.37 | 81.79±10.89 | **82.50±10.32** |
| | 70 | 82.50±4.36 | 83.57±6.01 | 83.21±5.06 | **84.64±4.24** |
| Sonar | 30 | 59.29±17.07 | 59.29±17.07 | **72.47±9.81** | 70.38±14.84 |
| | 50 | 65.49±15.70 | 65.49±15.70 | 73.29±9.05 | **76.37±12.49** |
| | 70 | 68.90±7/09 | 68.19±7.16 | 78.35±10.63 | **80.55±11.96** |
| Pima | 30 | 73.53±5.09 | 73.13±4.95 | **74.13±4.57** | 74.13±4.48 |
| | 50 | 75.54±7.06 | 76.54±7.08 | 77.53±5.70 | **77.54±7.17** |
| | 70 | 77.14±4.06 | 76.35±3.67 | 76.94±4.38 | **77.15±4.13** |
| win/draw/loss | | 26/1/3 | 25/1/4 | 21/1/8 | |

The scale of known labeled samples is significant in a semi-supervised framework. So, we evaluate the performance of Lap-PTSVM with the related model on scales ranging from known labeled samples 30%, 50%, and 70%. Tables 3 and 4 shows the mean and standard deviation of evaluation accuracy for the linear and nonlinear case, respectively. The win-draw-loss statistic is displayed in the table's bottom rows. Bold figures indicate the highest level of accuracy. Compared with

the results of Table 3, we establish that Lap-PTSVM are superior to that of TSVM, Pin-TSVM, and Lap-TSVM in 20 out of 30 situations (66.67%). Moreover in Tabel 4, Lap-PTSVM is also better than the other models in most of the datasets for non-linear case. This is demonstrated by the fact that 14 of the 21 datasets (66.67%) have the highest prediction accuracy. Here, it can be seen that adding structural information through the regularization term causes Lap-PTSVM to utilize all of the data available, helping the classifier to even better than the supervised model. Figure 20 confirms the results by comparing the average test performance of Pin-PTSVM, TSVM, Pin-TSVM, and Lap-TSVM over increased percentages of labeled data. It has been discovered that as the amount of labeled data increases, the total accuracy of all models improves. The selection of ineffective training points in some phases, which decreases performance, can be attributed to the fall in the learning curve.

In the previous section, we have shown that the Lap-PTSVM has noise insensitivity via the theoretical analysis. Now, we come to this property through experimentation. To even more clearly verify the model classification performance, we experiment with 12 binary classification datasets that are both noise-free and noisy, i.e., gradually adding noise to the data starting from $r = 0$ to $r = 0.1$. Tables 6 and 7 show the results of Lap-PTSVM and its related models in both linear and nonlinear cases, respectively. In terms of classification, Lap-PTSVM is comparable to other models in 21 out of 30 situations (70%) for linear case. As demonstrate in Table 6, Lap-PTSVM exceeds Lap-TSVM in almost datasets such as Banknote, WDBC, Heart-Stalog, Sonar and Breast. However, the accuracy of Lap-PTSVM is the same as the other models in Australian datasets and has lower accuracy than Lap-TSVM when $r = 0.05$ and 0.1 in Titanic datasets.

In the nonlinear case, comparable conclusions are obtained. Lap-PTSVM achieves better results for 18 out of 36 situations (50%). Although in some datasets, our model yields a lower accuracy than Lap-TSVM, it is better than TSVM and Pin-TSVM since Lap-PTSVM performing its training approach entirely on both labels and unlabeled data. In Banknote, Sonar, and Diabetes datasets, the proposed Lap-PTSVM has the highest accuracy in all cases. This is due to the fact that Lap-PTSVM using pinball loss construct a model that calculates error for all points,




(a) Appendicitis dataset

(b) Monk-3 dataset

(c) Spect dataset

(d) Sonar dataset

**Figure 20 Testing comparisons on the benchmark UCI dataset, the accuracy of several approaches with variable labeled samples (a) and (b) correspond to linear case , (c) and (d) correspond to nonlinear case.**

even if those points are properly classified. For this reason, Lap-PTSVM still produces good results when noise increase in both linear and nonlinear situation. Furthermore, the optimal parameters using in linear and nonlinear cases are shown in Tables 8 and 9, respectively.

**Table 4 The mean and standard deviation of tests accuracy with various labeled ratios on the UCI dataset were calculated using RBF kernel.**

| Datasets | Ratio | Existing model | | | Proposed model |
|---|---|---|---|---|---|
| | (%) | TSVM | Pin-TSVM | Lap-TSVM | Lap-PTSVM |
| Breast | 30 | 45.00±20.86 | 48.39±18.22 | 56.61±21.75 | **56.96±17.19** |
| | 50 | 61.79±15.57 | 59.29±12.67 | 56.79±19.70 | **62.32±14.21** |
| | 70 | **70.71±18.14** | 66.42±16.21 | 67.5±12.13 | 64.64±9.74 |
| Heart-Statlog | 30 | 73.88±11.28 | 71.44±7.18 | 76.05±6.43 | **77.16±7.19** |
| | 50 | 78.26±111 | 69.67±7.41 | 74.28±8.52 | **78.79±8.63** |
| | 70 | 84.05±6.63 | 70.81±6.63 | 82.81±8.65 | **85.16±6.40** |
| Ionosphere | 30 | 90.71±4.43 | **92.03±4.43** | 85.94±10.99 | 88.53±4.90 |
| | 50 | 90.75±5.97 | 91.64±5.97 | 92.96±6.23 | **93.40±5.27** |
| | 70 | 90.06±5.11 | 92.08±5.11 | **94.29±4.80** | 93.39±4.90 |
| Heart-C | 30 | 76.02±9.27 | **78.08±11.22** | 73.07±7.46 | 74.97±8.29 |
| | 50 | 77.52±9.61 | 78.03±9.61 | 79.13±9.80 | **80.11±8.78** |
| | 70 | 79.37±13.69 | 78.87±12.27 | **79.97±9.66** | 79.97±8.44 |
| Monk-2 | 30 | 91.07±7.36 | **91.07±7.70** | 83.57±7.18 | 82.14±13.55 |
| | 50 | **97.14±1.43** | 96.07±3.73 | 89.64±3.73 | 93.93±6.59 |
| | 70 | 96.78±2.97 | **97.71±3.11** | 92.86±6.19 | 97.50±2.79 |
| Spectheart | 30 | 79.67±9.83 | **83.10±11.44** | 79.67±9.83 | **83.10±11.44** |
| | 50 | 79.74±6.16 | 84.31±6.95 | 79.74±6.16 | **84.87±7.07** |
| | 70 | 79.74±5.50 | 82.12±11.41 | 79.74±5.50 | **83.86±10.02** |
| Sonar | 30 | 76.32±11.78 | 76.32±11.78 | 74.07±13.21 | **76.87±12.56** |
| | 50 | 80.77±8.73 | 81.48±8.82 | 79.95±12.19 | **82.80±12.37** |
| | 70 | 80.66±9.60 | 82.80±9.68 | 81.26±14.36 | **85.00±12.53** |
| win/draw/loss | | 16/0/5 | 14/1/6 | 17/0/4 | |

We also show that choosing hyper-parameter has an impact on levels of accuracy for 4 distinct datasets in Figure 21. The figure depicts 3D surface plots of accuracy versus noise and $\tau$. Other parameters are determined to the same optimal value as those in Table 8 and 9. Because of the pinball loss function, the proposed Lap-PTSVM improves results for noise-corrupted datasets. We investigate the performance with different noise level. Modifications in the hyper-parameter $\tau$ take a role in influencing accuracy; in this test, $\tau$ is set to various values: 0.01, 0.25, 0.5, 0.75, and 1, to demonstrate the shifting tendency of prediction accuracy. Even if the noise level rises, we are able to determine the $\tau$ that provided us with

the required accuracy.

### 3.1.8 Statistical analysis

The Friedman Test with post hoc test [34] is used to determine the statistical significance of the proposed Lap-PTSVM in comparison to TSVM, Pin-TSVM, and Lap-TSVM. The best performing algorithm is assigned to the smallest rank value in the Friedman test, which ranks each dataset separately. To show the efficiency of our proposed considering various noises, we compute the average rank of the Tables 6 and 7 as shown in Table 5.

For linear case, under the null hypothesis, which states that all the algorithms are equivalent. From the Table 5, the Friedman statistic is

$$\begin{aligned}\chi_F^2 &= \frac{12 \times 30}{4(4+1)}\left[3.62^2 + 2.92^2 + 2.05^2 + 1.42^2 - \frac{4(4+1)^2}{4}\right] \\ &= 50.44, \\ F_F &= \frac{(30-1) \times 50.44}{30 \times (4-1) - 50.44} = 36.95.\end{aligned}$$

We see that the critical value of $F(3, 87)$ at the confidence level of 0.05 is 2.723 and $36.95 > 2.723$, so the null hypothesis is rejected. This imply that there are significant difference among the four algorithms. From the Nemenyi post hoc test, the critical difference is $CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}$. Thus we can have $CD = 0.856$. As can be shown, the proposed Lap-PTSVM outperforms both TSVM and Pin-TSVM. However, the Nemenyi test fail to detect the significant difference between the proposed Lap-PTSVM and Lap-TSVM, but the proposed Lap-PTSVM has a lower average rank than the Lap-TSVM.

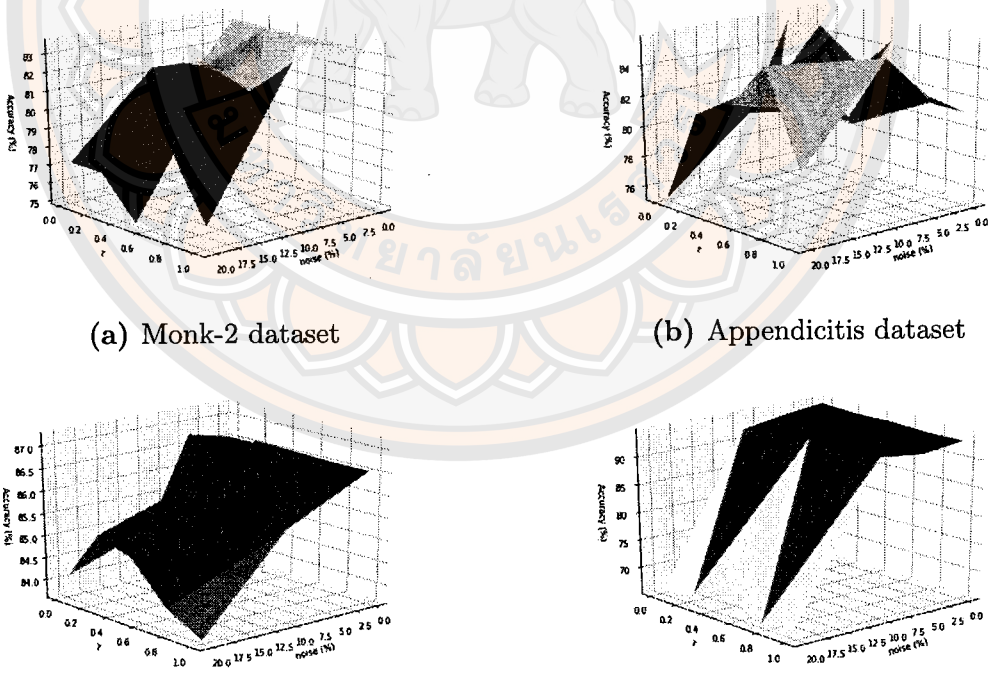
**Table 5 Friedman test of the models.**

| Kernel | TSVM | Pin-TSVM | Lap-TSVM | Lap-PTSVM |
|---|---|---|---|---|
| Linear kernel | 3.62 | 2.92 | 2.05 | 1.42 |
| RBF Kernel | 3.22 | 2.89 | 2.24 | 1.65 |

For nonlinear case, we also consider from Table 5, the Friedman statistic is

$$\chi_F^2 = \frac{12 \times 36}{4(4+1)} \left[ 3.22^2 + 2.89^2 + 2.24^2 + 1.65^2 - \frac{4(4+1)^2}{4} \right]$$
$$= 31.55,$$
$$F_F = \frac{(36-1) \times 31.55}{30 \times (4-1) - 31.55} = 14.44.$$

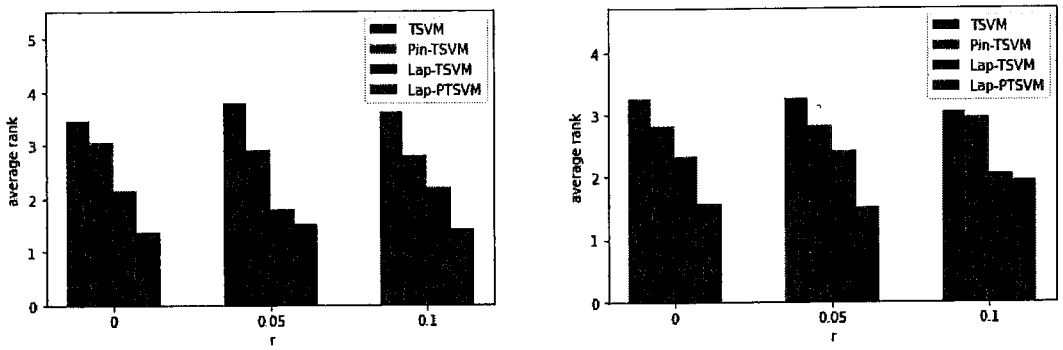
The critical value of $F(3, 105)$ at the confidence level of 0.05 is 2.699 and $14.44 > 2.699$, so the null hypothesis is rejected. Thus there are significant difference all over the four algorithms. Form the Nemenyi post hoc test, the critical difference is $CD = 0.7817$, so it fails to detect the significant difference between the proposed Lap-PTSVM and Lap-TSVM. However, the average rank of Lap-PTSVM is lowest. Furthermore, depending on the accuracy, the average ranks of all models with different percent of noise in both linear and nonlinear cases are computed and shown in Figure 22.



**(a)** Monk-2 dataset **(b)** Appendicitis dataset

**(c)** Australian dataset **(d)** Ionosphere dataset

**Figure 21 The variation in tests accuracy as a result of noise and $\tau$.**




(a) Linear case (b) Nonlinear case

**Figure 22 The average rank at different noise levels of each models (a) linear case (b) nonlinear case.**

### 3.1.9 Credit Approval Dataset

Commercial banks receive numerous requests for credit cards. However, many are rejected for various reasons, such as substantial loan amounts, low-income levels, or too many inquiries on an applicant's credit history. This dataset is interesting since it has a decent mix of features, including continuous, nominal with few values, and nominal with many values (https://www.kaggle.com). It consist of 690 samples with 15 attributes.

Firstly, we process the data and divide it into 70% unlabeled data and 30% labeled data. And then, tune other parameters except for $\tau$ and apply 10-fold cross-validation to validate our performance of Lap-PTSVM. Furthermore, we set the value of $\tau$ in the various values $(0.01, 0.1, 0.2, 0.5, 0.7, 1)$ to show the impact of pinball loss function on classification results when noise increases. Both linear and nonlinear situations are explored, as illustrated in Table 10 and 11, respectively.

**Table 6 The mean and standard deviation of tests accuracy with various noise on the UCI dataset were calculated using a linear kernel.**

| Datasets | r | Existing model | | | Proposed model |
|---|---|---|---|---|---|
| | | TSVM | Pin-TSVM | Lap-TSVM | Lap-PTSVM |
| Australian | 0 | **86.50±4.31** | **86.50±4.31** | **86.50±4.31** | **86.50±4.31** |
| | 0.05 | 72.11±16.19 | **86.78±4.59** | 86.78±4.13 | 86.56±4.31 |
| | 0.1 | 51.03±10.50 | 61.54±11.72 | **87.02±3.74** | 86.78±4.59 |
| Bupa | 0 | 60.71±9.22 | 62.02±6.94 | 57.51±13.39 | **66.96±11.91** |
| | 0.05 | 50.40±9.85 | 62.04±8.43 | 52.19±6.25 | **64.27±5.94** |
| | 0.1 | 53.62±10.71 | **59.76±8.26** | 53.52±10.78 | 58.46±8.33 |
| Ionosphere | 0 | 78.42±6.01 | 79.76±7.06 | **81.52±8.98** | 79.31±5.49 |
| | 0.05 | 75.81±9.76 | 75.81±8.73 | **82.82±6.95** | 78.87±5.03 |
| | 0.1 | 74.49±6.85 | 74.90±8.70 | 74.92±11.17 | **77.13±6.37** |
| Banknote | 0 | 45.39±3.50 | 71.64±4.29 | 96.45±1.54 | **98.26±0.01** |
| | 0.05 | 45.03±4.24 | 68.91±3.71 | 90.16±2.15 | **93.98±2.54** |
| | 0.1 | 44.93±3.89 | 67.91±2.96 | 82.96±4.28 | **88.07±3.34** |
| Pima | 0 | 75.54±7.06 | 76.54±7.08 | 77.54±5.70 | **77.54±7.18** |
| | 0.05 | 57.55±6.93 | 75.55±6.76 | **77.14±6.09** | 76.73±6.21 |
| | 0.1 | 73.53±6.46 | **73.93±5.83** | 73.33±6.26 | 73.54±5.80 |
| WDBC | 0 | 46.49±6.95 | 66.29±6.25 | 86.63±4.55 | **95.57±3.45** |
| | 0.05 | 45.62±5.46 | 65.20±7.53 | 86.75±6.21 | **93.59±3.89** |
| | 0.1 | 50.05±6.35 | 68.94±6.58 | 81.00±9.42 | **92.28±7.77** |
| Titanic | 0 | 64.43±8.94 | 64.37±11.71 | 75.53±2.95 | **77.53±2.95** |
| | 0.05 | 32.57±3.48 | 53.01±4.81 | **77.57±2.80** | 75.57±3.25 |
| | 0.1 | 32.73±3.91 | 53.81±4.66 | **77.57±2.79** | 70.91±2.83 |
| Heart-Statlog | 0 | 62.42±10.32 | 62.29±7.52 | 77.06±10.83 | **78.24±10.95** |
| | 0.05 | 59.48±11.20 | 60.03±11.67 | 77.65±10.25 | **81.11±9.12** |
| | 0.1 | 60.62±12.51 | 60.55±13.82 | 78.86±7.67 | **82.25±6.69** |
| Sonar | 0 | 65.49±15.70 | 65.49±15.70 | 73.29±9.05 | **76.37±12.49** |
| | 0.05 | 64.01±16.33 | 64.01±16.33 | 74.78±9.14 | **78.46±9.89** |
| | 0.1 | 68.35±17.49 | 68.35±17.49 | 71.09±7.79 | **74.47±12.68** |
| Breast | 0 | 55.54±13.37 | 55.54±13.37 | 63.57±16.12 | **67.86±11.49** |
| | 0.05 | 55.54±13.37 | 55.54±13.37 | 58.04±20.05 | **62.14±18.55** |
| | 0.1 | 55.54±13.37 | 55.54±13.37 | 59.64±12.78 | **66.79±12.14** |
| win/draw/loss | | 29/1/0 | 25/1/4 | 22/1/7 | |

**Table 7 The mean and standard deviation of tests accuracy with various noise on the UCI dataset were calculated using RBF kernel.**

| Datasets | r | Existing model | | | Proposed model |
|---|---|---|---|---|---|
| | | TSVM | Pin-TSVM | Lap-TSVM | Lap-PTSVM |
| Australian | 0 | 83.65±4.66 | 83.42±4.56 | 85.89±4.07 | **86.17±4.01** |
| | 0.05 | 84.32±4.54 | 83.87±4.72 | 85.44±4.09 | **85.89±4.63** |
| | 0.1 | 82.78±3.28 | 82.78±3.28 | **85.66±3.70** | 85.00±3.23 |
| Bupa | 0 | 48.59±10.79 | 52.62±9.52 | 57.96±8.19 | **66.87±7.31** |
| | 0.05 | 49.47±8.31 | 50.77±9.36 | 57.96±8.19 | **60.69±9.63** |
| | 0.1 | 50.41±9.57 | 50.88±9.92 | **57.96±8.19** | 53.93±14.51 |
| Ionosphere | 0 | 93.39±4.89 | 93.39±4.89 | 81.91±8.48 | **93.39±6.25** |
| | 0.05 | 92.05±3.82 | 92.05±3.82 | 79.25±7.17 | **92.53±4.81** |
| | 0.1 | **92.05±3.29** | **92.05±3.29** | 70.88±6.38 | 91.64±5.30 |
| Banknote | 0 | 96.85±1.31 | 96.97±1.34 | 98.43±1.24 | **99.77±0.44** |
| | 0.05 | 95.17±1.74 | 95.51±1.58 | 93.15±2.11 | **95.51±1.95** |
| | 0.1 | 81.70±4.27 | 83.28±2.87 | 85.85±4.34 | **88.20±3.46** |
| Pima | 0 | 68.91±7.52 | 70.72±7.48 | **76.34±6.50** | 72.13±5.87 |
| | 0.05 | 69.32±7.77 | 69.91±7.16 | **74.73±7.22** | 69.92±6.64 |
| | 0.1 | 67.69±10.00 | 68.09±10.08 | **70.70±8.36** | 66.12±6.83 |
| Heart-Statlog | 0 | 70.26±8.03 | 70.26±8.03 | **74.18±9.11** | 73.66±8.19 |
| | 0.05 | 72.58±8.67 | 72.58±8.67 | **74.77±7.96** | 73.63±7.86 |
| | 0.1 | 73.10±7.48 | 73.10±7.48 | **79.31±7.53** | 75.36±7.39 |
| Sonar | 0 | 79.23±12.79 | 79.23±12.79 | 79.94±12.19 | **81.31±11.07** |
| | 0.05 | 74.73±8.94 | 74.73±8.94 | 76.92±10.91 | 79.23±10.79 |
| | 0.1 | 69.78±11.30 | 69.78±11.30 | 76.26±15.72 | **80.05±9.87** |
| Breast | 0 | 63.39±12.45 | **64.82±10.71** | 55.53±13.37 | 63.57±15.11 |
| | 0.05 | 58.21±16.15 | 58.21±16.15 | 54.11±10.54 | **62.32±21.02** |
| | 0.1 | 44.64±14.11 | 44.64±14.11 | **54.11±10.54** | 50.18±19.68 |
| Diabetes | 0 | 65.95±7.13 | 67.35±6.74 | 65.09±8.49 | **73.52±7.95** |
| | 0.05 | 63.15±7.52 | 65.75±7.06 | 65.09±8.49 | **71.32±6.75** |
| | 0.1 | 65.54±5.98 | 65.35±6.55 | 65.09±8.49 | **68.91±5.87** |
| Fertility | 0 | 84.28±13.39 | 84.28±13.39 | **89.28±15.10** | 87.62±14.75 |
| | 0.05 | 84.28±13.39 | 84.28±13.39 | **89.28±15.10** | 85.95±14.18 |
| | 0.1 | 84.28±13.39 | 84.28±13.39 | **89.28±15.10** | **89.28±15.10** |
| Monk-2 | 0 | **95.00±3.98** | **95.00±3.98** | 70.71±12.45 | 90.71±5.10 |
| | 0.05 | 91.07±3.29 | **91.42±3.27** | 70.36±10.60 | 86.78±5.54 |
| | 0.1 | 82.50±7.04 | 82.50±7.04 | 73.57±13.09 | **83.57±6.62** |
| Spectheart | 0 | 77.94±6.03 | 77.94±6.03 | **79.73±6.15** | 79.15±7.19 |
| | 0.05 | 76.67±9.21 | 76.67±9.21 | **79.73±6.15** | 79.15±6.16 |
| | 0.1 | 78.49±5.16 | 78.49±5.16 | **79.73±6.15** | 79.15±6.16 |
| win/draw/loss | | 32/0/4 | 31/0/5 | 21/1/14 | |

**Table 8 The optimal parameters in linear case.**

| Datasets | TSVM | Pin-TSVM | Lap-TSVM | Lap-PTSVM |
|---|---|---|---|---|
| | $c_1, c_2$ | $c_1, c_2, \tau$ | $c_1, c_2, c_3, k$ | $c_1, c_2, c_3, \tau, k$ |
| Appendicitis | 1, 2 | 1, 2, 1 | 0.0625, 0.0625, 0.0625, 7 | 2, 0.25, 0.0625, 1, 4 |
| Australian | 1, 8 | 0.25, 8, 1 | 0.0625, 0.0625, 0.0625, 7 | 0.0625, 0.0625, 0.0625, 0.5, 7 |
| Breast | 4, 0.125 | 1, 0.0625, 0.5 | 4, 0.0625, 0.0625, 7 | 1, 0.125, 0.0625, 0.5, 7 |
| Bupa | 16, 4 | 16, 8, 0.5 | 16, 8, 1, 4 | 16, 4, 0.0625, 1, 4 |
| Heart-Statlog | 1, 8 | 0.25, 8, 1 | 1, 8 , 1, 4 | 0.25, 1, 0.0625, 1, 4 |
| Ionosphere | 4, 8 | 4, 8, 1 | 1, 8, 1, 4 | 1, 8, 0.0625, 1, 4 |
| Monk-2 | 0.25, 0.25 | 0.25, 0.25, 1 | 0.0625, 0.0625, 0.0625, 1 | 0.125, 0.0625, 0.0625, 1, 4 |
| Monk-3 | 0.5, 0.25 | 0.25, 0.5, 1 | 0.25, 0.5, 0.0625, 7 | 0.5, 0.25, 0.0625, 1, 4 |
| Sonar | 0.0625, 0.25 | 0.125, 0.5, 0.5 | 0.0625, 4, 0.0625, 4 | 0.125, 4, 0.0625, 1, 4 |
| Pima | 0.25, 0.5 | 0.25, 0.5, 0.5 | 0.25, 0.5, 0.125, 4 | 0.0626, 0.5, 0.0625, 1, 7 |
| WDBC | 1, 0.25 | 1, 16, 1 | 0.0625, 1, 0.0625, 4 | 0.25, 0.0625, 0.0625, 0.5, 4 |
| Titanic | 4, 16 | 1, 1, 1 | 0.0625, 16, 0.0625, 4 | 16, 0.0625, 0.0625, 1, 4 |
| Banknote | 1, 16 | 1, 4, 1 | 0.0625, 4, 0.0625, 4 | 16, 4, 0.0625, 0.5, 4 |

**Table 9 The optimal parameters in nonlinear case.**

| Datasets | TSVM | Pin-TSVM | Lap-TSVM | Lap-PTSVM |
|---|---|---|---|---|
| | $c_1, c_2, \sigma$ | $c_1, c_2, \tau, \sigma$ | $c_1, c_2, c_3,$ $k, \sigma$ | $c_1, c_2, c_3,$ $\tau, k, \sigma$ |
| Breast | 16, 1, 1 | 4, 16, 1, 0.1 | 16, 1, 1, 4, 1 | 16, 1 ,0.0625, 1, 4, 1 |
| Heart-Statlog | 0.25, 0.25, 0.1 | 0.0625, 0.25, 0.5, 0.01 | 0.0625, 1, 0.0625, 4, 1 | 0.25, 0.25, 0.25, 0.5, 7, 0.1 |
| Ionosphere | 0.0625, 0.0625, 0.1 | 0.0625, 0.625, 1, 0.1 | 2, 4, 4, 7, 1 | 4, 4, 4, 1, 7, 1 |
| Heart-C | 0.25, 0.25, 0.1 | 0.25, 0.25, 1, 0.1 | 4, 0.0625, 0.0625, 7, 1 | 4, 0.0625, 0.0625, 1, 7, 0.1 |
| Monk-2 | 16, 1, 1 | 16, 1, 1, 1 | 16, 1, 1, 4, 1 | 16, 1, 1, 0.25, 4, 0.1 |
| Spectheart | 0.0625, 0.25, 0.01 | 0.25, 0.0625, 0.5, 10 | 0.0625, 0.0625, 0.25, 7, 0.01 | 0.25, 0.0625, 0.0625, 1, 7, 10 |
| Sonar | 4, 4, 0.1 | 4, 4, 1, 0.1 | 4, 4, 0.0625, 7, 0.1 | 4, 4, 0.0625, 1, 7, 0.1 |
| Australian | 0.0625, 0.0625, 0.01 | 0.0625, 0.0625, 0.5, 0.01 | 0.0625, 0.0625, 0.0625, 4, 0.01 | 0.0625, 0.5, 0.0625, 0.5, 4, 0.01 |
| Bupa | 0.25, 0.0625, 0.01 | 0.125, 0.0625, 0.5, 0.01 | 0.0625, 0.0625, 0.0625, 4, 0.1 | 0.0625, 0.0625, 0.0625, 0.5 0.5, 4, 0.1 |
| Diabetes | 0.0625, 0.0625, 0.1 | 0.0625, 0.0625, 0.5, 0.1 | 0.0625, 0.0625, 0.0625, 4, 0.1 | 0.0625, 0.0625,0.0625, 0.5, 4, 0.01 |
| Fertility | 0.0625, 0.0625, 0.01 | 0.0625, 0.0625, 0.5, 0.01 | 0.0625, 0.0625, 0.0625, 4, 0.01 | 0.0625, 0.0625, 0.0625, 0.5, 4, 0.01 |
| Banknote | 1, 8, 0.01 | 2, 8, 0.5, 0.01 | 0.0625, 0.25, 0.0625, 4, 0.01 | 0.25, 0.0625, 4, 0.5, 4, 0.01 |

**Table 10 The mean and standard deviation of tests accuracy with various noise and $\tau$ on the Credit approval dataset using linear kernel.**

| Model | Noise | | |
|---|---|---|---|
| | 0 | 0.05 | 0.1 |
| TSVM | 50.58±7.98 | 50.07±7.22 | 51.11±7.56 |
| Pin-TSVM | 76.48±7.46 | 76.22±9.57 | 71.27±5.39 |
| Lap-TSVM | 79.66±8.28 | 78.30±8.19 | 73.59±7.09 |
| Lap-PTSVM ($\tau = 0.01$) | 79.91±7.32 | 78.07±6.51 | 77.30±5.00 |
| Lap-PTSVM ($\tau = 0.1$) | 79.65±7.40 | 78.34±7.21 | 77.31±4.04 |
| Lap-PTSVM ($\tau = 0.2$) | 79.14±7.67 | 78.09±5.44 | 76.49±5.49 |
| Lap-PTSVM ($\tau = 0.5$) | 80.71±7.58 | 78.33±8.44 | **78.09±7.42** |
| Lap-PTSVM ($\tau = 0.7$) | **81.23±7.37** | **79.41±9.28** | 75.70±6.22 |
| Lap-PTSVM ($\tau = 1$) | 80.97±7.47 | 79.39±9.23 | 77.59±7.02 |

One can see that, Our Lap-PTSVM get the highest accuracy in all of the case, and the almost higher score compared to the others model belongs to Lap-PTSVM corresponding to different value of $\tau$.

**Table 11 The mean and standard deviation of tests accuracy with various noise and $\tau$ on the Credit approval dataset using RBF kernel.**

| Model | Noise | | |
|---|---|---|---|
| | 0 | 0.05 | 0.1 |
| TSVM | 76.74±7.80 | 78.83±7.72 | 76.46±17.38 |
| Pin-TSVM | 75.71±6.16 | 78.59±6.30 | 73.64±5.77 |
| Lap-TSVM | 82.28±5.92 | 80.72±6.42 | 78.87±6.46 |
| Lap-PTSVM ($\tau = 0.01$) | 83.83±5.40 | 81.48±4.79 | 80.95±5.68 |
| Lap-PTSVM ($\tau = 0.1$) | 83.83±5.40 | 83.02±5.37 | 80.43±4.05 |
| Lap-PTSVM ($\tau = 0.2$) | 84.34±5.53 | 83.04±5.71 | **83.58±6.95** |
| Lap-PTSVM ($\tau = 0.5$) | **84.35±5.91** | **83.58±6.43** | 80.68±4.68 |
| Lap-PTSVM ($\tau = 0.7$) | **84.35±5.91** | 82.77±6.70 | 79.13±5.42 |
| Lap-PTSVM ($\tau = 1$) | **84.35±5.91** | 83.56±7.80 | 81.47±7.34 |

## 3.2 Improved laplacian twin support vector machine based on generalized pinball loss

Motivate by the idea of GPin-TSVM and Lap-TSVM, we introduce the model call LapGPin-TSVM. This extension aims to transition from supervised learning to semi-supervised learning by incorporating the Laplacian Regularization Term.

### 3.2.1 Primal problem

In the classification problem, our goal is to define two non-parallel hyperplanes similar to the equations provided in (2.7.4). In terms of the problem formulation, we can express it as an optimization problem incorporating the generalized pinball loss, as follows:

$$\begin{aligned}\min_{\mathbf{w}_1,b_1,\boldsymbol{\xi}} \quad & \frac{1}{2}\|A\mathbf{w}_1 + e_1 b_1\|^2 + c_1\mathbf{e}_2^\top\boldsymbol{\xi} + \frac{c_2}{2}(\|\mathbf{w}_1\|^2 + b_1^2) \\ & + \frac{c_3}{2}(M\mathbf{w}_1 + \mathbf{e}b_1)^\top L(M\mathbf{w}_1 + \mathbf{e}b_1) \\ \text{s.t.} \quad & -(B\mathbf{w}_1 + \mathbf{e}_2 b_1) \geq \mathbf{e}_2 - \frac{1}{\tau_1}(\boldsymbol{\xi} + \mathbf{e}_2\epsilon_1), \\ & -(B\mathbf{w}_1 + \mathbf{e}_2 b_1) \leq \mathbf{e}_2 + \frac{1}{\tau_2}(\boldsymbol{\xi} + \mathbf{e}_2\epsilon_2), \\ & \boldsymbol{\xi} \geq 0,\end{aligned} \tag{3.2.1}$$

and

$$\begin{aligned}\min_{\mathbf{w}_2,b_2,\boldsymbol{\xi}} \quad & \frac{1}{2}\|B\mathbf{w}_2 + \mathbf{e}_2 b_2\|^2 + c_1\mathbf{e}_1^\top\boldsymbol{\xi}_1 + \frac{c_2}{2}(\|\mathbf{w}_2\|^2 + b_2^2) \\ & + \frac{c_3}{2}(M\mathbf{w}_2 + \mathbf{e}b_2)^\top L(M\mathbf{w}_2 + \mathbf{e}b_2) \\ \text{s.t.} \quad & (A\mathbf{w}_2 + \mathbf{e}_1 b_2) \geq \mathbf{e}_1 - \frac{1}{\tau_3}(\boldsymbol{\xi} + \mathbf{e}_1\epsilon_3), \\ & (A\mathbf{w}_2 + \mathbf{e}_1 b_2) \leq \mathbf{e}_1 + \frac{1}{\tau_4}(\boldsymbol{\xi} + e_1\epsilon_4), \\ & \boldsymbol{\xi} \geq 0,\end{aligned} \tag{3.2.2}$$

where $c_1, c_2, c_3$ are non-negative parameters and $\tau_1, \tau_2, \tau_3, \tau_4, \epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4 \geq 0$.

In problem (3.2.1), the first term aims to minimize the sum of the squared distances of labeled samples to the hyperplane, while the second term represents the

slack variable controlling the loss of samples by employing the concept of generalized pinball loss. The term $\|\mathbf{w}_1\|^2+b_1^2$ serves as regularization to prevent ill-conditioning. The fourth term is the Laplacian regularization term, introducing a penalty for deviations from smoothness in the decision function across the data manifold. It encourages the model to respect the underlying geometric structure of the data, promoting a more coherent decision boundary.

The given minimization problem can be converted into an unconstrained optimization format by interpreting the constraint as the significance of generalized pinball loss. This leads to the formulation of the new problem as follows:

$$\begin{aligned}\min_{\mathbf{w}_1,b_1}\ & \frac{1}{2}\|A\mathbf{w}_1+\mathbf{e}_1b_1\|^2+c_1\mathbf{e}_2^\top\mathcal{L}_{\tau_1,\tau_2}^{\epsilon_1,\epsilon_2}\big(\mathbf{e}_2+(B\mathbf{w}_1+\mathbf{e}_2b_1)\big)+c_2(\|\mathbf{w}_1\|^2+b_1^2)\\ & +c_3(\mathbf{w}_1^\top M^\top+\mathbf{e}^\top b_1)L(M\mathbf{w}_1+\mathbf{e}b_1),\end{aligned}\tag{3.2.3}$$

and

$$\begin{aligned}\min_{\mathbf{w}_2,b_2}\ & \frac{1}{2}\|B\mathbf{w}_2+\mathbf{e}_2b_2\|^2+c_1\mathbf{e}_1^\top\mathcal{L}_{\tau_1,\tau_2}^{\epsilon_1,\epsilon_2}\big(\mathbf{e}_1-(A\mathbf{w}_2+\mathbf{e}_1b_2)\big)+c_2(\|\mathbf{w}_2\|^2+b_2^2)\\ & +c_3(\mathbf{w}_2^\top M^\top+\mathbf{e}^\top b_2)L(M\mathbf{w}_2+\mathbf{e}b_2).\end{aligned}\tag{3.2.4}$$

These problems may yield different representations of the constrained optimization (3.2.1) and (3.2.2), but they have equivalent solutions to the original problems.

**Remark 3.2.1.** If we set $c_2, c_3 = 0$ in (3.2.1) and (3.2.2), these problems degenerate to GPin-TSVM.

In seeking the solution to the optimization problems (3.2.1) and (3.2.2), we reformulate it into dual forms and utilize quadratic programming problems. For considering the dual form and solving the problem, we focus on problem (3.2.1), as the computation method for the problem (3.2.2) is the same. Here, we introduce Lagrange multiplier $\boldsymbol{\alpha},\boldsymbol{\gamma},\boldsymbol{\beta}\geq\mathbf{0}$ and then obtain the following form:

$$\begin{aligned}L(\mathbf{w}_1,b_1,\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\gamma},\boldsymbol{\beta}) =&\frac{1}{2}\|A\mathbf{w}_1+\mathbf{e}_1b_1\|^2+c_1\mathbf{e}_2^\top\boldsymbol{\xi}+\frac{c_2}{2}(\|\mathbf{w}\|^2+b^2)\\ &+\frac{c_3}{2}(M\mathbf{w}_1+\mathbf{e}b_1)^\top L(M\mathbf{w}_1+\mathbf{e}b_1)\\ &-\boldsymbol{\alpha}^\top(-(B\mathbf{w}_1+\mathbf{e}_2b_1)-\mathbf{e}_2+\frac{1}{\tau_1}(\boldsymbol{\xi}+\mathbf{e}_2\epsilon_1))\end{aligned}$$

$$- \boldsymbol{\gamma}^{\top}(\mathbf{e}_2 + \frac{1}{\tau_2}(\boldsymbol{\xi} + \mathbf{e}_2\epsilon_2) + (B\mathbf{w}_1 + \mathbf{e}_2 b_1)) - \boldsymbol{\beta}^{\top}\boldsymbol{\xi}. \tag{3.2.5}$$

Next, we find the subdifferential of the above and obtain:

$$\frac{\partial L}{\partial \mathbf{w}_1} = A^{\top}(A\mathbf{w}_1 + \mathbf{e}_1 b_1) + c_2\mathbf{w}_1 + c_3 M^{\top}L(M\mathbf{w}_1 + \mathbf{e}b_1) + \boldsymbol{\alpha}^{\top}B - \boldsymbol{\gamma}^{\top}B = 0, \tag{3.2.6}$$

$$\frac{\partial L}{\partial b_1} = \mathbf{e}_1^{\top}(A\mathbf{w}_1 + \mathbf{e}_1 b_1) + c_2 b_1 + c_3\mathbf{e}^{\top}L(M\mathbf{w}_1 + \mathbf{e}b_1) + \boldsymbol{\alpha}^{\top}\mathbf{e}_2 - \boldsymbol{\gamma}^{\top}\mathbf{e}_2 = 0, \tag{3.2.7}$$

$$\frac{\partial L}{\partial \boldsymbol{\xi}} = c_1\mathbf{e}_2 - \frac{\boldsymbol{\alpha}}{\tau_1} - \frac{\boldsymbol{\gamma}}{\tau_2} - \boldsymbol{\beta} = 0, \tag{3.2.8}$$

$$- \boldsymbol{\alpha}^{\top}(-(B\mathbf{w}_1 + \mathbf{e}_2 b_1) - \mathbf{e}_2 + \frac{1}{\tau_1}(\boldsymbol{\xi} + \mathbf{e}_2\epsilon_1)) = 0, \tag{3.2.9}$$

$$- \boldsymbol{\gamma}^{\top}(\mathbf{e}_2 + \frac{1}{\tau_2}(\boldsymbol{\xi} + \mathbf{e}_2\epsilon_2) + (B\mathbf{w}_1 + \mathbf{e}_2 b_1)) = 0, \tag{3.2.10}$$

$$- \boldsymbol{\beta}^{\top}\boldsymbol{\xi} = 0. \tag{3.2.11}$$

Since $\boldsymbol{\beta} \geq 0$, we have

$$c_1\mathbf{e}_2 - \frac{\boldsymbol{\alpha}}{\tau_1} - \frac{\boldsymbol{\gamma}}{\tau_2} = \boldsymbol{\beta} \geq 0, \tag{3.2.12}$$

which implies that

$$c_1\mathbf{e}_2 \geq \frac{\boldsymbol{\alpha}}{\tau_1} + \frac{\boldsymbol{\gamma}}{\tau_2}. \tag{3.2.13}$$

Now, we define $F = \begin{bmatrix} B & \mathbf{e}_2 \end{bmatrix}$, $H = \begin{bmatrix} A & \mathbf{e}_1 \end{bmatrix}$, $J = \begin{bmatrix} M & \mathbf{e} \end{bmatrix}$, $z_1 = \begin{bmatrix} \mathbf{w}_1 & b_1 \end{bmatrix}^{\top}$ and $z_2 = \begin{bmatrix} \mathbf{w}_2 & b_2 \end{bmatrix}^{\top}$ Then, the dual form problem of (3.2.1) is

$$\begin{aligned} \min_{\boldsymbol{\alpha},\boldsymbol{\gamma}} \ & \frac{1}{2}(\boldsymbol{\alpha} - \boldsymbol{\gamma})^{\top}F(H^{\top}H + c_2 I + c_3 J^{\top}LJ)^{-1}F^{\top}(\boldsymbol{\alpha} - \boldsymbol{\gamma}) + (\boldsymbol{\alpha} - \boldsymbol{\gamma})^{\top}\mathbf{e}_2(1 + \frac{\epsilon_2}{\tau_2}) \\ & - \boldsymbol{\alpha}^{\top}\mathbf{e}_2(\frac{\epsilon_1}{\tau_1} + \frac{\epsilon_2}{\tau_2}) \\ \text{s.t.} \ & \frac{\boldsymbol{\alpha}}{\tau_1} + \frac{\boldsymbol{\gamma}}{\tau_2} \leq c_1\mathbf{e}_2, \\ & \boldsymbol{\alpha},\ \boldsymbol{\gamma} \geq 0. \end{aligned} \tag{3.2.14}$$

After solving the quadratic programming problem, the solution is obtain and then we get

$$z_1 = -(H^{\top}H + c_1 I + c_3 J^{\top}LJ)^{-1}F^{\top}(\boldsymbol{\alpha} - \boldsymbol{\gamma}). \tag{3.2.15}$$

Similar to the negative hyperplane, we obtain

$$\begin{aligned}\min_{\boldsymbol{\mu},\boldsymbol{\eta}} \ & \frac{1}{2}(\boldsymbol{\mu}-\boldsymbol{\eta})^\top H(F^\top F + c_5 I + c_6 J^\top L J)^{-1} H^\top(\boldsymbol{\mu}-\boldsymbol{\eta}) + (\boldsymbol{\mu}-\boldsymbol{\eta})^\top \mathbf{e}_1(1+\frac{\epsilon_4}{\tau_4}) \\ & - \boldsymbol{\mu}^\top \mathbf{e}_1(\frac{\epsilon_3}{\tau_3}+\frac{\epsilon_4}{\tau_4}) \\ \text{s.t.} \ & \frac{\boldsymbol{\mu}}{\tau_3}+\frac{\boldsymbol{\eta}}{\tau_4} \leq c_4\mathbf{e}_1, \\ & \boldsymbol{\mu},\ \boldsymbol{\eta} \geq 0. \end{aligned} \tag{3.2.16}$$

The solution of this problem is

$$z_2 = (F^\top F + c_5 I + c_6 J^\top L J)^{-1} H^\top(\boldsymbol{\mu}-\boldsymbol{\eta}). \tag{3.2.17}$$

Similarly with the previous work, we provide the Python code. However, if you intend to employ the code, we recommend reviewing it once more because the code presented here originates from the Class (BaseEstimator), resulting in the presence of self in the function input.

```
import numpy as np
from sklearn.base import BaseEstimator
from sklearn.neighbors import kneighbors_graph
from scipy.sparse import *
from cvxopt import matrix,solvers
import math
```

In this work, we did not build the function construct laplacian independently; instead, we generated the Laplacian graph in the code concurrently with the process of solving QPPs. Here, the Python code

```
    def fit(self,X,Y):
        self.X = X
        self.Y = Y
        #set seed of data
        np.random.seed(8)
        #generate data into label and unlabeled data
        ind_0 = np.nonzero(Y == -1)[0]
        ind_1 = np.nonzero(Y == 1)[0]
        Y_pos = Y[Y==1]
        Y_neg = Y[Y==-1]
        n_label_pos = math.floor(len(Y_pos) - (self.percent_unlabel
            *len(Y_pos)))
```

```
12          n_label_neg = math.floor(len(Y_neg) - (self.percent_unlabel
               *len(Y_neg)))
13          ind_l0=np.random.choice(ind_0,n_label_neg,False)
14          ind_u0=np.setdiff1d(ind_0,ind_l0)
15          ind_l1 = np.random.choice(ind_1,n_label_pos, False)
16          ind_u1 = np.setdiff1d(ind_1, ind_l1)
17          #labeled data for each class
18          A = X[ind_l1,:]  #label 1
19          B = X[ind_l0,:]  #label -1
20          #unlabeled data for each class
21          Au = X[ind_u1,:]
22          Bu = X[ind_u0,:]
23          #all of unlabeled data
24          Xu = np.vstack([Au, Bu])
25          #construct_laplacian
26          knn_dist_graph = kneighbors_graph(X=self.X,
27                                            n_neighbors=self.n_neighbor,
28                                            mode='distance',
29                                            metric='euclidean',
30                                            n_jobs=6)
31          sigma = 4
32          similarity_graph = lil_matrix(knn_dist_graph.shape)
33          nonzeroindices = knn_dist_graph.nonzero()
34          #calculate weight of data
35          similarity_graph[nonzeroindices] = np.exp((-1)*np.asarray(
               knn_dist_graph[nonzeroindices])**2 / 2.0 * sigma**2)
36          similarity_graph = 0.5 * (similarity_graph +
               similarity_graph.T)
37          degree_matrix = similarity_graph.sum(axis=1)
38          diagonal_matrix = np.diag(np.asarray(degree_matrix).reshape
               (X.shape[0],))
39          L =  diagonal_matrix - similarity_graph  #L=D-A
40          self.L = L
41          total = X.shape[0]
```

Now, we have positive labeled data, negative labeled data, and unlabeled data, along with the Laplacian graph, ready for the next step. We then transform problems (3.2.14) and (3.2.16) into a format that can be solved by the QPPs. The code runs after the previous one as follows.

```
#get number of each class
l_A = A.shape[0]
l_B = B.shape[0]
#construct matrix H (positive data)
e1 = matrix(np.ones((l_A, 1)))
H = matrix(np.concatenate((A, e1), axis=1))
 #construct matrix F (negative data)
e2 = matrix(np.ones((l_B, 1)))
F = matrix(np.concatenate((B, e2), axis=1))
 #construct matrix J (both label and unlabeled data)
t = matrix(np.ones((total, 1)))
J = matrix(np.concatenate((self.X, t), axis=1))
#construct (H.TH+c_2I+c_3J.TLJ)^-1
s1 = matrix(np.dot(H.T,H) + self.c_2*(np.identity(np.size(H
    ,1))) + self.c_3*np.dot(np.dot(J.T,L),J))
d1 = matrix(np.linalg.inv(s1))
self.d1 = d1
del s1
' Set the problem into the form of solver.qpp '
#consider positive hyperplane
#construct P
P1_11 = matrix(np.dot(np.dot(F,d1),F.T))
P1_12 = matrix(np.zeros(shape=(len(B),len(B))))
P1_21 = matrix(np.zeros(shape=(len(B),len(B))))
P1_22 = matrix(np.zeros(shape=(len(B),len(B))))
P1_r1 = matrix(np.concatenate((P1_11, P1_12), axis=1))
P1_r2 = matrix(np.concatenate((P1_21, P1_22), axis=1))
del P1_11, P1_12, P1_21, P1_22
P1 = matrix(np.concatenate((P1_r1, P1_r2)))
#construct q
q1_11 = matrix((1+(self.eps_2/self.tau_2))*np.ones((len(B)
    ,1)))
q1_21 = matrix(((self.eps_1/self.tau_1)+(self.eps_2/self.
    tau_2))*(-1)*np.ones(len(B)))
q1 = matrix(np.concatenate((q1_11, q1_21)))
del q1_11, q1_21
#construct G
G1_11 = matrix((-1/self.tau_2)*np.identity(len(B)))
G1_12 = matrix(((1/self.tau_1)+(1/self.tau_2))*np.identity(
    len(B)))
```

```
37          G1_21 = matrix(np.zeros((len(B),len(B))))
38          G1_22 = matrix((-1)*np.identity(len(B)))
39          G1_31 = matrix(np.identity(len(B)))
40          G1_32 = matrix((-1)*np.identity(len(B)))
41
42          G1_r1 = matrix(np.concatenate((G1_11, G1_12), axis=1))
43          G1_r2 = matrix(np.concatenate((G1_21, G1_22), axis=1))
44          G1_r3 = matrix(np.concatenate((G1_31, G1_32), axis=1))
45          del G1_11, G1_12, G1_21, G1_22, G1_31, G1_32,
46          G1 = matrix(np.concatenate((G1_r1, G1_r2, G1_r3)))
47          del G1_r1, G1_r2, G1_r3
48          #construct h
49          h1_r1 = matrix(self.c_1*np.ones((len(B),1)))
50          h1_r2 = matrix(np.zeros((len(B),1)))
51          h1_r3 = matrix(np.zeros((len(B),1)))
52          # h1_r4 = matrix(np.zeros((len(B),1)))
53          h1 = matrix(np.concatenate((h1_r1, h1_r2, h1_r3)))
54          del h1_r1,h1_r2, h1_r3
55          ' Solving QPP using the package solvers.qpp '
56          sol_1 = solvers.qp(P1,q1,G1,h1)
57          del P1, q1, G1, h1
58          alpha_gam = sol_1['x'][0:l_B]
59          del sol_1
60          #solution
61          z_1 = (-1)*np.dot(np.dot(d1,F.T),alpha_gam)
62          del alpha_gam
```

Next is the code to obtain the solution for the negative hyperplane.

```
1           #using H, F and J from above
2           s2 = matrix(np.dot(F.T,F) + self.c_2*(np.identity(np.size(F
                ,1)))+ self.c_3*np.dot(np.dot(J.T,L),J))
3           del L
4           d2 = matrix(np.linalg.inv(s2))
5           del s2
6           ' Set the problem into the form of solvers.qp'
7           #construct P
8           P2_11 = matrix(np.dot(np.dot(H,d2),H.T))
9           P2_12 = matrix(np.zeros(shape=(len(A),len(A))))
10          P2_21 = matrix(np.zeros(shape=(len(A),len(A))))
```

```
11          P2_22 = matrix(np.zeros(shape=(len(A),len(A))))
12          P2_r1 = matrix(np.concatenate((P2_11, P2_12), axis=1))
13          P2_r2 = matrix(np.concatenate((P2_21, P2_22), axis=1))
14          del P2_11, P2_12, P2_21, P2_22
15          P_neg = matrix(np.concatenate((P2_r1, P2_r2)))
16          del P2_r1, P2_r2
17          #construct q
18          q2_11 = matrix((1+(self.eps_2/self.tau_2))*np.ones((len(A)
                ,1)))
19          q2_21 = matrix((-1)*((self.eps_1/self.tau_1)+(self.eps_2/
                self.tau_2))*np.ones((len(A),1)))
20          q_neg = matrix(np.concatenate((q2_11, q2_21)))
21          del q2_11, q2_21
22          #construct G
23          G2_11 = matrix((-1/self.tau_2)*np.identity(len(A)))
24          G2_12 = matrix(((1/self.tau_1)+(1/self.tau_2))*np.identity(
                len(A)))
25          G2_21 = matrix(np.zeros((len(A),len(A))))
26          G2_22 = matrix((-1)*np.identity(len(A)))
27          G2_31 = matrix(np.identity(len(A)))
28          G2_32 = matrix((-1)*np.identity(len(A)))
29
30          G2_r1 = matrix(np.concatenate((G2_11, G2_12), axis=1))
31          G2_r2 = matrix(np.concatenate((G2_21, G2_22), axis=1))
32          G2_r3 = matrix(np.concatenate((G2_31, G2_32), axis=1))
33          del G2_11, G2_12, G2_21, G2_22, G2_31, G2_32
34          G_neg = matrix(np.concatenate((G2_r1, G2_r2, G2_r3)))
35          del G2_r1, G2_r2, G2_r3
36          #construct h
37          h2_31 = matrix(self.c_2*np.ones((len(A),1)))
38          h2_32 = matrix(np.zeros((len(A),1)))
39          h2_33 = matrix(np.zeros((len(A),1)))
40          h_neg = matrix(np.concatenate((h2_31, h2_32, h2_33)))
41          del h2_31, h2_32, h2_33
42          #Solving negative class
43          sol_neg = solvers.qp(P_neg,q_neg,G_neg,h_neg)
44          del P_neg, q_neg, G_neg, h_neg
45          x_neg = sol_neg['x']
46          del sol_neg
47          beta_pho = x_neg[0:l_A]
```

```
48          #solution
49          z_2 = matrix(np.dot(np.dot(d2,H.T),beta_pho))
50          del x_neg, beta_pho
```

After obtaining the solutions for the two QPPs, we can retrieve $w_1, b_1$ and $w_2, b_2$ using the following code.

```
1       #the optimal solution of positive hyperplane
2       w_1 = z_1[0:-1]
3       b_1 = z_1[-1]
4       #the optimal solution of negative hyperplane
5       w_2 = z_2[0:-1]
6       b_2 = z_2[-1]
7       self.w_1 = w_1
8       self.b_1 = b_1
9       self.w_2 = w_2
10      self.b_2 = b_2
11      return self
```

After acquiring the two hyperplanes, we categorize the new data sample $x_i$ using the following expression:

$$\text{class}(i) = \arg\min_{i=1,2} \frac{|\mathbf{x}^\top \mathbf{w}_i + b_i|}{\|\mathbf{w}_i\|}. \tag{3.2.18}$$

In this context, $|.|$ represents the perpendicular distance of the data from the hyperplane.

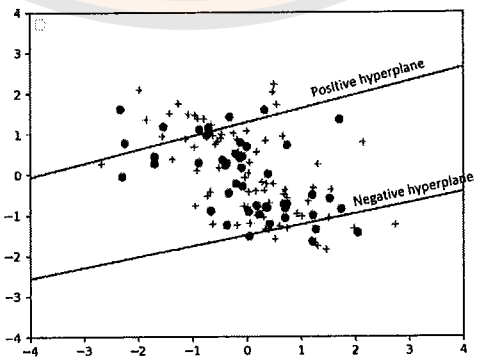



**Figure 23 The two hyperplanes obtained from LapGPin-TSVM.**

### 3.2.2 Noise Insensitivity

We explore how LapGPin-TSVM tackles the problem of sensitivity to noise. To be concise, we concentrate on the linear case and resolve the issue presented in (3.2.3). However, it is worth noting that the same analysis is applicable to the nonlinear case. The separating hyperplane obtained from our proposed method is shown in Figure 23. Define the generalized sign function $\mathrm{sgn}^{\epsilon_1,\epsilon_2}_{\tau_1,\tau_2}(v)$ as

$$\mathrm{sgn}^{\epsilon_1,\epsilon_2}_{\tau_1,\tau_2}(v) = \begin{cases} \tau_1 & \text{if } v > \frac{\epsilon_1}{\tau_1}, \\ [0,\tau_1] & \text{if } v = \frac{\epsilon_1}{\tau_1}, \\ 0 & \text{if } -\frac{\epsilon_2}{\tau_2} < v < \frac{\epsilon_1}{\tau_1}, \\ [-\tau_2,0] & \text{if } v = -\frac{\epsilon_2}{\tau_2}, \\ -\tau_2 & \text{if } v < -\frac{\epsilon_2}{\tau_2}, \end{cases} \tag{3.2.19}$$

where $v = 1 - y(\mathbf{w}^\top \mathbf{x} + b)$ and $\mathrm{sgn}^{\epsilon_1,\epsilon_2}_{\tau_1,\tau_2}(v)$ represents the subgradient of the generalized pinball loss function. Similar to theprocess of Lap-PTSVM. Consider the equation (3.2.3), we can formulate it as follows:

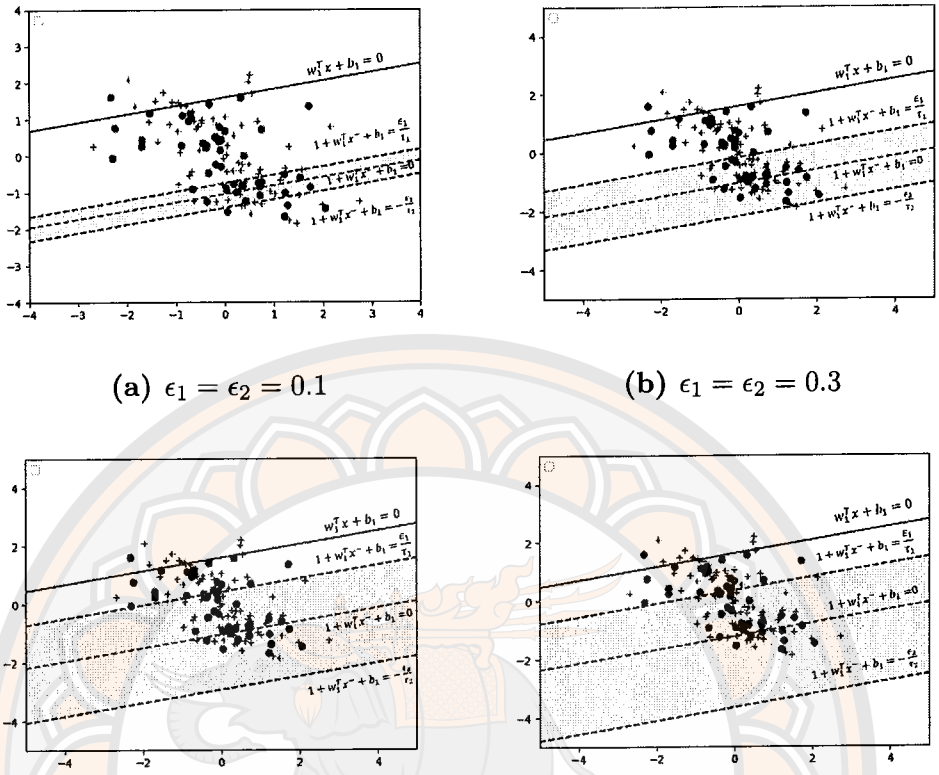
$$\begin{aligned} \mathbf{0} \in c_1 \sum_{i=1}^{m_2} \mathrm{sgn}^{\epsilon_1,\epsilon_2}_{\tau_1,\tau_2}(1 + (\mathbf{w}_1 \mathbf{x}_i^- + b_1))\mathbf{x}_i^- + c_2 \mathbf{w}_1 + c_3 M^\top L(M\mathbf{w}_1 + \mathbf{e}b_1) \\ + A^\top (A\mathbf{w}_1 + \mathbf{e}_1 b_1). \end{aligned} \tag{3.2.20}$$

Here, $\mathbf{0}$ represents the zero vector, $\mathbf{x}_i^- \in B$, and $m_2$ is the number of negative samples.

Given $w_1$ and $b_1$, we can partition the index set of $B$ into five distinct sets:

$$\begin{aligned} V_1^+ &= \{i : 1 + (\mathbf{w}_1^\top \mathbf{x}_i^- + b_1) > \frac{\epsilon_1}{\tau_1}\}, \\ V_2^+ &= \{i : 1 + (\mathbf{w}_1^\top \mathbf{x}_i^- + b_1) = \frac{\epsilon_1}{\tau_1}\}, \\ V_3^+ &= \{i : -\frac{\epsilon_2}{\tau_2} < 1 + (\mathbf{w}_1^\top \mathbf{x}_i^- + b_1) < \frac{\epsilon_1}{\tau_1}\}, \\ V_4^+ &= \{i : 1 + (\mathbf{w}_1^\top \mathbf{x}_i^- + b_1) = -\frac{\epsilon_2}{\tau_2}\}, \\ V_5^+ &= \{i : 1 + (\mathbf{w}_1^\top \mathbf{x}_i^- + b_1) < -\frac{\epsilon_2}{\tau_2}\}, \end{aligned}$$

where, $i \in \{1, 2, 3, \ldots, m_2\}$. By introducing the notation $V_1^+, V_2^+, V_3^+, V_4^+$, and $V_5^+$, Equation (3.2.20) can be reformulated to assert the existence of $\phi_i \in [0, \tau_1]$ and



**(a)** $\epsilon_1 = \epsilon_2 = 0.1$ **(b)** $\epsilon_1 = \epsilon_2 = 0.3$

**(c)** $\epsilon_1 = \epsilon_2 = 0.5$ **(d)** $\epsilon_1 = 0.6, \epsilon_2 = 0.5$

**Figure 24 Illustration the set $V_3^+$ (the shadow area) when using different of $\epsilon_1, \epsilon_2$.**

$\theta_i \in [-\tau_2, 0]$ for which:

$$\tau_1 \sum_{i \in V_1^+} \mathbf{x}_i^- + \sum_{i \in V_2^+} \phi_i \mathbf{x}_i^- + \sum_{i \in V_4^+} \theta_i \mathbf{x}_i^- - \tau_2 \sum_{i \in V_5^+} \mathbf{x}_i^- +$$
$$\frac{1}{c_1} A^\top (A\mathbf{w}_1 + \mathbf{e}_1 b_1) + \frac{c_2}{c_1} \mathbf{w}_1 + \frac{c_3}{c_1} M^\top L(M\mathbf{w}_1 + \mathbf{e} b_1) = \mathbf{0}.$$

As indicated in Equation (3.2.19), the samples in $V_3^+$ may not contribute to the improvement of $\mathbf{w}_1$ due to the fact that the generalized sign function is zero. However, $V_3^+$ directly influences the sparsity of the model. It is important to note that the quantities $\epsilon_1$ and $\epsilon_2$ regulate the number of samples in $V_3^+$. As $\epsilon_1$ and $\epsilon_2$ approach zero, sparsity diminishes. Conversely, when $\epsilon_1 \to \infty$ and $\epsilon_2 \to \infty$, we enhance sparsity by including a greater number of samples in $V_3^+$. The conclusion are shown in the Figure 24.

**Proposition 3.2.2.** If there exists a solution to the optimization problem (3.2.1), the following inequalities must be satisfied:

$$-\frac{1}{c_1\tau_1 m_2}[\mathbf{e}_1^\top(A\mathbf{w}_1+\mathbf{e}_1 b_1)+c_2 b_1+c_3\mathbf{e}^\top L(M\mathbf{w}_1+\mathbf{e}b_1)]\leq 1,$$

and

$$\frac{p_1}{m_2}\leq 1-\frac{\tau_1+\frac{\mathbf{e}_1^\top(A\mathbf{w}_1+\mathbf{e}_1 b_1)+c_2 b_1+c_3\mathbf{e}^\top L(M\mathbf{w}_1+\mathbf{e}b_1)}{c_1 m_2}}{\tau_1+\tau_2},$$

where $p_1$ denote the number samples in $V_1^+$.

*Proof.* Consider an arbitrary negative point, denoted as $\mathbf{x}_{i_0}^-$, belonging to the set $V_1^+$. Utilizing the KKT conditions represented by equations (3.2.10) and (3.2.11), we derive that $\beta_{i_0}=\gamma_{i_0}=0$. Further analysis of the KKT condition (3.2.8) leads to the conclusion that $\alpha_{i_0}=c_1\tau_1$, resulting in $\alpha_{i_0}-\gamma_{i_0}=c_1\tau_1$.

Define $\boldsymbol{\lambda}=\boldsymbol{\alpha}-\boldsymbol{\gamma}$, and consequently, $\lambda_{i_0}=c_1\tau_1$. Additionally, the expression $-\big(\mathbf{e}_1^\top(A\mathbf{w}_1+\mathbf{e}_1 b_1)+c_2 b_1+c_3\mathbf{e}^\top L(M\mathbf{w}_1+\mathbf{e}b_1)\big)=p_1c_1\tau_1+\sum_{i\notin E_1^+}\lambda_i$ is obtained from (3.2.7).

Considering the constraints $\alpha_i\geq 0$ and $\gamma_i\geq 0$, it follows that $-c_1\tau_2\leq\lambda_i\leq c_1\tau_1$. Thus, the sum of $\lambda_i$ over points not in $V_1^+$ is given by

$$\sum_{i\notin V_1^+}\lambda_i=-[\mathbf{e}_1^\top(A\mathbf{w}_1+\mathbf{e}_1 b_1)+c_2 b_1+c_3\mathbf{e}^\top L(M\mathbf{w}_1+\mathbf{e}b_1)]-p_1c_1\tau_1.$$

Consequently, we establish that $-c_1\tau_2\leq\frac{\sum_{i\notin V_1^+}\lambda_i}{m_2-p_1}\leq c_1\tau_1$. This leads to the inequality:

$$-[\mathbf{e}_1^\top(A\mathbf{w}_1+\mathbf{e}_1 b_1)+c_2 b_1+c_3\mathbf{e}^\top L(M\mathbf{w}_1+\mathbf{e}b_1)]-(m_2-p_1)c_1\tau_1\leq p_1c_1\tau_1,$$

and

$$p_1c_1\tau_1\leq-[\mathbf{e}_1^\top(A\mathbf{w}_1+\mathbf{e}_1 b_1)+c_2 b_1+c_3\mathbf{e}^\top L(M\mathbf{w}_1+\mathbf{e}b_1)]+(m_2-p_1)c_1\tau_1.$$

This completes the proof of our argument. □

When analyzing the aforementioned proposition, it finds that the value of $\tau_1,\tau_2$ affect the number of samples in the set $V_1^+$. Decreasing the $\tau_1,\tau_2$ value

results in fewer members in $V_1^+$, causing the decision boundary to be more sensitive to noise. Conversely, increasing the $\tau_1, \tau_2$ value make the decision boundary less sensitive to noise. Additionally, there is a term related to the graph-based approach, namely the Laplacian term. This introduces the consideration that the analysis of both labeled and unlabeled data influences the creation of the classification boundary.

In considering the negative hyperplane, we define a set in a similar manner but with respect to positive samples instead. These set are as follows:

$$
\begin{aligned}
V_1^- &= \{i : 1 - (\mathbf{w}_2^\top \mathbf{x}_i^+ + b_2) > \frac{\epsilon_3}{\tau_3}\}, \\
V_2^- &= \{i : 1 - (\mathbf{w}_2^\top \mathbf{x}_i^+ + b_2) = \frac{\epsilon_3}{\tau_3}\}, \\
V_3^- &= \{i : -\frac{\epsilon_4}{\tau_4} < 1 - (\mathbf{w}_2^\top \mathbf{x}_i^+ + b_2) < \frac{\epsilon_3}{\tau_3}\}, \\
V_4^- &= \{i : 1 - (\mathbf{w}_2^\top \mathbf{x}_i^+ + b_2) = -\frac{\epsilon_4}{\tau_4}\}, \\
V_5^- &= \{i : 1 - (\mathbf{w}_2^\top \mathbf{x}_i^+ + b_2) < -\frac{\epsilon_4}{\tau_4}\},
\end{aligned}
$$

Where, $i \in \{1, 2, 3, \ldots, m_1\}$. For the analysis, we use the same approach. Consequently, we obtain the following proposition.

**Proposition 3.2.3.** If there exists a solution to the optimization problem (3.2.2), the following inequalities must be satisfied:

$$
\frac{1}{c_1 \tau_3 m_1}[\mathbf{e}_2^\top (B\mathbf{w}_2 + \mathbf{e}_2 b_2) + c_2 b_2 + c_3 \mathbf{e}^\top L(M\mathbf{w}_2 + \mathbf{e}b_2)] \leq 1,
$$

and

$$
\frac{p_2}{m_1} \leq 1 - \frac{\tau_3 + \frac{\mathbf{e}_2^\top (B\mathbf{w}_2 + \mathbf{e}_2 b_2) + c_2 b_2 + c_3 \mathbf{e}^\top L(M\mathbf{w}_2 + \mathbf{e}b_2)}{c_1 m_1}}{\tau_3 + \tau_4},
$$

where $p_2$ denote the number samples in $V_1^-$.

The inclusion of Laplacian regularization in LapGPin-TSVM contributes to enhanced generalization and robustness, particularly in scenarios where the data exhibits intrinsic geometric properties.

**Table 12 The detailed description of 13 benchmark datasets.**

| Datasets | No. of Sample | No. of Feature |
|---|---|---|
| Ionosphere | 351 | 33 |
| Bupa | 345 | 6 |
| Fertility | 100 | 10 |
| Pima | 768 | 8 |
| Banknote | 1372 | 4 |
| Monk-2 | 432 | 7 |
| Sonar | 208 | 60 |
| Diabetes | 769 | 9 |
| Spambase | 4601 | 57 |
| WDBC | 569 | 30 |
| Australian | 690 | 14 |
| Heart | 303 | 13 |
| Specf heart | 267 | 22 |

### 3.2.3 Experiment Results

In this section, GPin-TSVM [18], Lap-TSVM [14], and Lap-PTSVM [39] are compared with LapGPin-TSVM. We selected 13 benchmark datasets to evaluate the performance of our proposed model. A comprehensive overview of these datasets is presented in Table 12. A Grid Search was employed to explore a range of hyperparameters. We fine-tuned the parameters $c_1$, $c_2$, and $c_3$ from the set $\{10^i, i = -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5\}$, while $\tau_1$, $\tau_2$, $\tau_3$, $\tau_4$, $\epsilon_1$, $\epsilon_2$, $\epsilon_3$, and $\epsilon_4$ were selected from the range $(0, 1)$. In the context of non-linear case, the kernel parameter $\sigma$ tuning from the set $\{10^i, i = -3, -2, -1, 0, 1, 2, 3\}$.

All experiments were executed in Python 3.9.5 on a Windows 10 system, utilizing an Intel(R) Core(TM) i7-4500U CPU @ 1.80GHz 2.40 GHz. The experimental results were obtained from 5-fold cross-validation. Our investigation focuses on the model performance, emphasizing the effects of varying ratios of unlabeled data and noise. The bold type indicates the best result.

**Table 13 The mean of ACC, MCC, and time with 20% unlabeled data.**

| Datasets | | GPin-TSVM<br>MCC<br>Time (s) | Lap-TSVM<br>MCC<br>Time (s) | Lap-PTSVM<br>MCC<br>Time (s) | LapGPin-TSVM<br>MCC<br>Time (s) |
|---|---|---|---|---|---|
| Fertility | Linear | **88.00±5.10** | **88.00±5.10** | **88.00±5.10** | **88.00±5.10** |
| | | **0.0500** | 0.000 | 0.000 | 0.000 |
| | | 0.0352 | 0.0361 | 0.0397 | 0.0404 |
| | RBF | **88.00±4.00** | **88.00±4.00** | **88.00±4.00** | **88.00±4.00** |
| | | **0.0000** | **0.000** | **0.000** | **0.000** |
| | | 0.0469 | 0.0421 | 0.0521 | 0.0570 |
| Banknote | Linear | 93.88±1.97 | **98.98±0.63** | 97.52±1.43 | 98.03±0.94 |
| | | 0.8804 | **0.9793** | 0.9516 | 0.9607 |
| | | 2.1492 | 0.6919 | 0.9516 | 0.9607 |
| | RBF | **100.00±0.00** | 99.78±0.18 | 99.85±0.29 | **100.00±0.00** |
| | | **1.000** | 0.9956 | 0.9971 | **1.000** |
| | | 2.2177 | 0.6225 | 2.6625 | 2.1435 |
| Bupa | Linear | 60.00±3.85 | **68.41±4.80** | 67.53±3.50 | 67.25±3.73 |
| | | 0.8804 | **0.9793** | 0.9516 | 0.9607 |
| | | 2.1492 | 0.6919 | 0.9516 | 0.9607 |
| | RBF | **68.69±7.25** | 66.67±4.80 | 67.53±3.50 | 67.25±3.73 |
| | | 0.8804 | **0.9793** | 0.9516 | 0.9607 |
| | | 2.1492 | 0.6919 | 0.9516 | 0.9607 |
| Ionosphere | Linear | 81.48±3.75 | 88.03±2.51 | 87.74±2.35 | **90.88±2.66** |
| | | 0.6207 | 0.7419 | 0.7365 | **0.8040** |
| | | 0.3662 | 0.1751 | 0.1499 | 0.1080 |
| | RBF | 89.19±3.94 | **92.89±4.21** | 91.18±3.11 | 91.18±2.74 |
| | | 0.7709 | **0.8443** | 0.8065 | 0.8068 |
| | | 0.1506 | 0.0732 | 0.1158 | 0.1205 |
| Monk-2 | Linear | 79.17±3.74 | 83.81±1.98 | **86.35±2.04** | 80.55±2.25 |
| | | 0.5914 | 0.6769 | **0.7278** | 0.6264 |
| | | 0.1253 | 0.0956 | 0.1670 | 0.1221 |
| | RBF | 96.29±3.71 | **97.22±3.33** | **97.22±3.33** | **97.22±3.33** |
| | | 0.9272 | **0.9464** | **0.9464** | **0.9464** |
| | | 0.1726 | 0.0986 | 0.1809 | 0.1586 |
| Pima | Linear | 68.36±2.76 | 75.39±1.41 | 75.39±1.41 | **76.95±1.06** |
| | | 0.2343 | 0.4251 | 0.4251 | **0.4687** |
| | | 0.5448 | 0.1411 | 0.6538 | 0.4987 |
| | RBF | 72.67±4.82 | 75.79±3.06 | **79.97±4.15** | 76.05±4.01 |
| | | 0.3663 | 0.4424 | **0.4716** | 0.4499 |
| | | 0.5341 | 0.1874 | 0.4813 | 0.5510 |

**Table 13 (Cont.)**

| | | | | | |
|---|---|---|---|---|---|
| Sonar | Linear | 75.45±9.58 | 77.86±6.07 | **78.36±6.25** | 77.86±6.54 |
| | | 0.5380 | 0.5585 | **0.5723** | 0.5605 |
| | | 0.0632 | 0.0688 | 0.0710 | 0.0704 |
| | RBF | 75.45±3.08 | 77.92±6.17 | **79.83±2.28** | 77.91±4.79 |
| | | 0.5185 | 0.5605 | **0.5955** | 0.5642 |
| | | 0.0552 | 0.0546 | 0.0779 | 0.0661 |
| Diabetes | Linear | 75.39±1.62 | 74.48±2.19 | **77.73±1.95** | 76.82±2.11 |
| | | 0.4350 | 0.4093 | **0.4928** | 0.4722 |
| | | 0.4728 | 0.1441 | 0.6376 | 0.5980 |
| | RBF | 75.77±2.03 | 76.43±1.47 | 76.82±1.63 | **76.95±1.13** |
| | | 0.4433 | 0.4611 | 0.4715 | **0.4744** |
| | | 0.5273 | 0.2047 | 0.6880 | 0.5473 |
| Spambase | Linear | 84.08±1.01 | 90.45±0.91 | **91.32±1.11** | 91.12±0.95 |
| | | 0.7106 | 0.8000 | **0.8177** | 0.8135 |
| | | 50.0703 | 16.4100 | 61.5567 | 42.6016 |
| | RBF | 88.41±0.97 | 90.30±1.09 | 91.28±0.57 | **91.41±0.51** |
| | | 0.7569 | 0.7971 | 0.8186 | **0.8194** |
| | | 41.5564 | 17.3560 | 79.4900 | 70.7805 |
| WDBC | linear | 88.93±1.17 | 92.96±1.86 | 94.90±1.52 | **95.78±1.41** |
| | | 0.7729 | 0,8594 | 0.8927 | **0.9106** |
| | | 0.2742 | 0.1037 | 0.3075 | 0.2995 |
| | RBF | 93.15±1.02 | 95.26±2.57 | 95.26±2.63 | **95.43±2.79** |
| | | 0.8597 | 0.8992 | 0.8991 | **0.9051** |
| | | 0.2595 | 0.1216 | 0.3249 | 0.3304 |
| Australian | linear | 64.35±3.79 | 85.65±4.21 | 85.94±4.36 | **86.09±3.95** |
| | | 0.3319 | 0.7192 | **0.7272** | 0.7265 |
| | | 0.3362 | 0.1195 | 0.3587 | 0.3109 |
| | RBF | 67.68±2.08 | 76.81±3.67 | **84.20±4.00** | 76.23±1.96 |
| | | 0.3676 | 0.5298 | **0.6865** | 0.5216 |
| | | 0.3553 | 0.1410 | 0.4298 | 0.5042 |
| Heart | linear | 75.56±5.90 | 81.85±4.59 | 84.07±3.01 | **84.81±2.16** |
| | | 0.5169 | 0.6301 | 0.6756 | **0.6905** |
| | | 0.0691 | 0.0630 | 0.0919 | 0.0879 |
| | RBF | 79.62±1.17 | 83.70±2.46 | 81.48±3.70 | **81.48±4.05** |
| | | 0.5923 | 0.6664 | **0.6317** | 0.6279 |
| | | 0.1039 | 0.0735 | 0.1030 | 0.0836 |
| Spect heart | linear | 77.16±3.16 | 77.53±3.90 | 79.03±3.78 | **79.41±3.91** |
| | | **0.1456** | 0.0212 | 0.0107 | 0.000 |
| | | 0.0691 | 0.0630 | 0.0919 | 0.0879 |
| | RBF | 76.42±4.45 | 77.19±5.04 | 81.62±3.92 | **82.01±4.84** |
| | | 0.0981 | 0.0547 | 0.2136 | **0.2701** |
| | | 0.0817 | 0.0651 | 0.1078 | 0.1036 |
| win/tile/loss | | 22/3/1 | 15/4/7 | 13/4/9 | |

**Table 14 The mean of ACC, MCC, and time with 40% unlabeled data.**

| Datasets | | GPin-TSVM<br>MCC<br>Time (s) | Lap-TSVM<br>MCC<br>Time (s) | Lap-PTSVM<br>MCC<br>Time (s) | LapGPin-TSVM<br>MCC<br>Time (s) |
|---|---|---|---|---|---|
| Fertility | Linear | 79.00±11.58 | 85.00±7.75 | **88.00±5.10** | **88.00±5.10** |
| | | **0.0111** | -0.0439 | 0.000 | 0.000 |
| | | 0.0339 | 0.0303 | 0.0365 | 0.0441 |
| | RBF | **88.00±4.00** | **88.00±4.00** | **88.00±4.00** | **88.00±4.00** |
| | | **0.0000** | **0.000** | **0.000** | **0.000** |
| | | 0.0376 | 0.0521 | 0.0472 | 0.0427 |
| Banknote | Linear | 94.31±1.74 | **98.76±0.68** | 97.45±1.36 | 98.17±0.73 |
| | | 0.8880 | **0.9749** | 0.9502 | 0.9635 |
| | | 1.1178 | 0.3627 | 1.1045 | 1.1202 |
| | RBF | 99.78±0.43 | 99.78±0.29 | 99.56±0.42 | **99.85±0.42** |
| | | 0.9956 | 0.9956 | 0.9912 | **0.9971** |
| | | 1.0669 | 0.4062 | 1.4074 | 1.1707 |
| Bupa | Linear | 62.60±5.53 | 65.21±5.50 | **65.22±6.28** | 64.93±6.18 |
| | | 0.2042 | **0.2718** | 0.2701 | 0.2636 |
| | | 0.0587 | 0.0572 | 0.0712 | 0.0680 |
| | RBF | 64.93±6.82 | 65.70±6.12 | **67.25±5.91** | 66.96±6.69 |
| | | 0.2809 | 0.2990 | **0.3407** | 0.3362 |
| | | 0.0694 | 0.0520 | 0.0876 | 0.0795 |
| Ionosphere | Linear | 82.04±2.82 | 86.60±3.36 | 88.03±2.17 | **89.17±4.30** |
| | | 0.6305 | 0.7067 | 0.7467 | **0.7660** |
| | | 0.1310 | 0.0682 | 0.1060 | 0.0849 |
| | RBF | 88.05±4.60 | 90.04±3.47 | **90.62±3.37** | 88.89±3.97 |
| | | 0.7402 | 0.7798 | **0.7904** | 0.7554 |
| | | 0.0838 | 0.0567 | 0.0895 | 0.0954 |
| Monk-2 | Linear | 80.57±3.85 | **84.73±1.31** | 84.27±2.74 | 84.24±4.66 |
| | | 0.6198 | 0.6928 | 0.6880 | **0.6954** |
| | | 0.0999 | 0.0587 | 0.1280 | 0.1051 |
| | RBF | 96.75±2.88 | 97.21±3.34 | **97.22±3.34** | **97.22±3.34** |
| | | 0.9331 | 0.9461 | **0.9464** | **0.9464** |
| | | 0.0935 | 0.0848 | 0.1213 | 0.1084 |
| Pima | Linear | 68.23±2.72 | 75.65±1.16 | 75.91±1.09 | **76.95±1.24** |
| | | 0.2272 | 0.4303 | 0.4375 | **0.4695** |
| | | 0.2582 | 0.1118 | 0.3220 | 0.2599 |
| | RBF | 72.41±4.14 | 75.66±3.24 | 75.66±4.13 | **75.92±3.77** |
| | | 0.3505 | 0.4328 | 0.4344 | **0.4429** |
| | | 0.2719 | 0.1302 | 0.2659 | 0.2874 |

**Table 14 (Cont.)**

| | | | | | |
|---|---|---|---|---|---|
| Sonar | Linear | 72.62±8.57 | 76.96±10.079 | **79.35±7.86** | 77.91±8.48 |
| | | 0.4729 | 0.5431 | **0.6057** | 0.5743 |
| | | 0.0542 | 0.0434 | 0.0577 | 0.0595 |
| | RBF | 73.53±7.69 | **75.53±4.40** | 73.10±5.53 | 75.47±4.67 |
| | | 0.4680 | **0.5192** | 0.4635 | 0.5154 |
| | | 0.0497 | 0.0543 | 0.0677 | 0.0586 |
| Diabetes | Linear | 73.57±1.92 | 73.83±2.41 | 76.69±1.37 | **77.08±1.76** |
| | | 0.3883 | 0.3966 | 0.4736 | **0.4801** |
| | | 0.2361 | 0.1030 | 0.3358 | 0.3023 |
| | RBF | 74.99±2.24 | 75.78±1.98 | 76.30±1.69 | **77.60±2.21** |
| | | 0.4218 | 0.4445 | 0.4599 | **0.4877** |
| | | 0.2482 | 0.1479 | 0.3597 | 0.2975 |
| Spambase | Linear | 84.10±1.36 | 88.82±0.36 | **90.86±1.17** | 90.78±1.05 |
| | | 0.7113 | 0.7656 | **0.8081** | 0.8062 |
| | | 23.1552 | 7.7955 | 26.8444 | 20.8474 |
| | RBF | 88.21±0.84 | 89.97±0.95 | 91.12±0.77 | **91.15±0.69** |
| | | 0.7534 | 0.7901 | 0.8135 | **0.8138** |
| | | 20.1447 | 8.5742 | 38.3108 | 38.6832 |
| WDBC | Linear | 88.40±2.02 | 93.84±2.69 | 94.72±1.48 | **95.78±1.03** |
| | | 0.7616 | 0.8734 | 0.8890 | **0.9099** |
| | | 0.1466 | 0.0845 | 0.1730 | 0.1735 |
| | RBF | 93.32±0.88 | 94.38±3.53 | 94.38±2.63 | **95.08±2.25** |
| | | 0.8627 | 0.8796 | 0.8794 | **0.8966** |
| | | 0.1378 | 0.1044 | 0.1781 | 0.0065 |
| Australian | linear | 64.49±3.58 | 85.36±4.45 | 85.80±3.57 | **86.09±3.38** |
| | | 0.3435 | 0.7098 | 0.7211 | **0.7217** |
| | | 0.1923 | 0.0912 | 0.1931 | 0.2084 |
| | RBF | 69.57±4.69 | 74.92±3.26 | **81.88±4.74** | 75.51±1.41 |
| | | 0.3791 | 0.4920 | **0.6426** | 0.5080 |
| | | 0.1821 | 0.1089 | 0.2568 | 0.2597 |
| Heart | linear | 74.81±8.65 | 80.74±5.93 | **83.33±5.10** | 83.33±4.96 |
| | | 0.5050 | 0.6071 | 0.6597 | **0.6610** |
| | | 0.0520 | 0.0493 | 0.0818 | 0.0719 |
| | RBF | 77.41±3.59 | 78.89±3.43 | 80.37±3.43 | **81.85±2.16** |
| | | 0.5667 | 0.5689 | 0.6072 | **0.6312** |
| | | 0.0777 | 0.0687 | 0.0886 | 0.0817 |
| Spect heart | linear | 78.27±2.26 | **80.16±2.38** | 79.02±4.49 | 79.03±3.22 |
| | | **0.2266** | 0.1670 | 0.0390 | 0.0231 |
| | | 0.0604 | 0.0521 | 0.0756 | 0.0707 |
| | RBF | 76.43±6.79 | 75.66±4.87 | 80.51±4.27 | **80.89±5.62** |
| | | 0.1111 | -0.0399 | 0.2659 | **0.2863** |
| | | 0.0638 | 0.0537 | 0.0734 | 0.0787 |
| win/tile/loss | | 25/1/0 | 19/1/6 | 15/3/8 | |

**Table 15 The mean of ACC, MCC, and time with 60% unlabeled data.**

| Datasets | | GPin-TSVM<br>MCC<br>Time (s) | Lap-TSVM<br>MCC<br>Time (s) | Lap-PTSVM<br>MCC<br>Time (s) | LapGPin-TSVM<br>MCC<br>Time (s) |
|---|---|---|---|---|---|
| Fertility | Linear | 75.00±10.95 | 85.00±8.94 | 86.00±4.90 | **88.00±5.10** |
| | | 0.0709 | **0.0882** | -0.0306 | 0.000 |
| | | 0.0228 | 0.0355 | 0.0372 | 0.0347 |
| | RBF | 88.00±4.00 | 88.00±4.00 | **89.00±3.74** | 88.00±4.00 |
| | | 0.0000 | 0.000 | **0.1092** | 0.000 |
| | | 0.0277 | 0.0382 | 0.0399 | 0.0467 |
| Banknote | Linear | 94.31±2.45 | **98.47±0.78** | 97.45±1.51 | 97.96±0.94 |
| | | 0.888 | **0.9690** | 0.9502 | 0.9591 |
| | | 0.3644 | 0.2067 | 0.4249 | 0.4271 |
| | RBF | 99.19±0.74 | 99.13±0.91 | 99.71±0.27 | **99.85±0.18** |
| | | 0.9838 | 0.9823 | 0.9941 | **0.9971** |
| | | 0.3521 | 0.2644 | 0.5851 | 0.4778 |
| Bupa | Linear | 60.28±5.53 | 65.21±5.50 | **65.22±6.28** | 64.93±6.18 |
| | | 0.2042 | **0.2718** | 0.2701 | 0.2636 |
| | | 0.0587 | 0.0572 | 0.0712 | 0.0680 |
| | RBF | 64.64±3.85 | 66.96±2.95 | 67.83±7.52 | **68.12±6.54** |
| | | 0.2566 | 0.3162 | 0.3405 | **0.3512** |
| | | 0.0657 | 0.0506 | 0.0608 | 0.0658 |
| Ionosphere | Linear | 79.77±3.57 | 86.61±1.47 | 87.45±3.57 | **87.75±5.00** |
| | | 0.5971 | 0.7098 | 0.7323 | **0.7385** |
| | | 0.2376 | 0.0538 | 0.0722 | 0.0612 |
| | RBF | 86.05±5.25 | 86.35±6.12 | 86.62±2.27 | **87.20±5.83** |
| | | 0.7010 | 0.6965 | 0.7054 | **0.7202** |
| | | 0.0653 | 0.0560 | 0.0627 | 0.0599 |
| Monk-2 | Linear | 78.71±3.02 | 78.25±3.10 | 78.95±2.12 | **84.04±3.13** |
| | | 0.5904 | 0.5601 | 0.5752 | **0.6840** |
| | | 0.0548 | 0.0504 | 0.0784 | 0.0755 |
| | RBF | 95.60±2.69 | **97.22±3.33** | 96.75±3.24 | **97.22±3.33** |
| | | 0.9102 | **0.9464** | 0.9373 | **0.9464** |
| | | 0.0617 | 0.0657 | 0.0878 | 0.0799 |
| Pima | Linear | 69.27±2.35 | 74.99±1.73 | 75.00±1.73 | **77.08±1.03** |
| | | 0.2693 | 0.4188 | 0.4188 | **0.4703** |
| | | 0.1095 | 0.0741 | 0.1552 | 0.1421 |
| | RBF | 73.32±5.46 | 75.14±3.78 | 75.66±3.54 | **75.92±4.19** |
| | | 0.3803 | 0.4232 | 0.4336 | **0.4423** |
| | | 0.1224 | 0.1209 | 0.1507 | 0.1578 |

## Table 15 (Cont.)

| | | | | | |
|---|---|---|---|---|---|
| Sonar | Linear | 66.91±12.22 | **74.56±5.99** | 71.72±8.78 | 72.68±9.93 |
| | | 0.3398 | **0.5016** | 0.4567 | 0.4740 |
| | | 0.0399 | 0.0543 | 0.0481 | 0.0452 |
| | RBF | 66.36±3.54 | **71.65±2.61** | 68.75±4.78 | 69.24±8.48 |
| | | 0.3353 | **0.4334** | 0.3879 | 0.3932 |
| | | 0.0350 | 0.0402 | 0.0469 | 0.0461 |
| Diabetes | Linear | 73.96±2.18 | 73.83±2.18 | 76.17±1.89 | **76.56±2.28** |
| | | 0.3983 | 0.3938 | 0.4572 | **0.4685** |
| | | 0.1118 | 0.0800 | 0.1773 | 0.1562 |
| | RBF | 76.30±1.99 | 76.69±1.25 | 76.95±2.50 | **77.08±1.79** |
| | | 0.4556 | 0.4695 | 0.4770 | **0.4823** |
| | | 0.1293 | 0.1041 | 0.1670 | 0.1552 |
| Spambase | Linear | 84.60±1.15 | 88.77±1.47 | 90.82±1.13 | **90.88±1.01** |
| | | 0.7177 | 0.7645 | 0.8069 | **0.8084** |
| | | 7.8185 | 3.3341 | 9.7254 | 7.8930 |
| | RBF | 87.32±0.90 | 89.45±1.28 | **91.28±0.71** | 91.02±0.93 |
| | | 0.7347 | 0.7792 | **0.8168** | 0.8113 |
| | | 7.2114 | 3.4492 | 12.8823 | 11.6014 |
| WDBC | Linear | 85.94±2.02 | 94.19±0.89 | 94.73±2.15 | **95.43±1.69** |
| | | 0.7110 | 0.8783 | 0.8892 | **0.9023** |
| | | 0.1014 | 0.0632 | 0.1276 | 0.1188 |
| | RBF | 93.49±1.05 | **94.38±2.51** | 94.20±2.45 | 94.03±2.30 |
| | | 0.8666 | **0.8806** | 0.8761 | 0.8750 |
| | | 0.0735 | 0.0836 | 0.1322 | 0.1245 |
| Australian | linear | 65.07±4.48 | 84.05±4.85 | 85.22±4.24 | **85.94±4.04** |
| | | 0.3513 | 0.6841 | 0.7110 | **0.7216** |
| | | 0.1013 | 0.0742 | 0.1169 | 0.1065 |
| | RBF | 69.27±4.68 | 74.78±2.56 | **78.12±4.62** | 73.48±2.85 |
| | | 0.3754 | 0.4888 | **0.5657** | 0.4656 |
| | | 0.1071 | 0.0860 | 0.1388 | 0.1552 |
| Heart | linear | 72.96±5.32 | 81.11±5.90 | **82.96±4.28** | 82.22±5.19 |
| | | 0.4608 | 0.6166 | **0.6534** | 0.6396 |
| | | 0.0572 | 0.0556 | 0.0638 | 0.0525 |
| | RBF | 76.67±2.51 | 80.00±1.39 | 80.00±0.74 | **80.00±3.59** |
| | | 0.5680 | 0.5952 | 0.6002 | **0.6034** |
| | | 0.0587 | 0.0540 | 0.0685 | 0.0645 |
| Spect heart | linear | 74.89±3.92 | 77.54±2.23 | **78.65±4.02** | 78.65±3.47 |
| | | **0.2415** | 0.1317 | 0.0952 | 0.0139 |
| | | 0.0394 | 0.0378 | 0.0685 | 0.0584 |
| | RBF | 75.65±3.52 | 77.51±4.68 | **79.43±3.94** | 77.18±4.46 |
| | | 0.0578 | 0.1535 | **0.2400** | 0.2137 |
| | | 0.0648 | 0.0439 | 0.0552 | 0.0544 |
| win/tile/loss | | 25/1/0 | 17/2/7 | 18/0/8 | |

**Table 16 The mean of ACC, MCC, and time with 80% unlabeled data.**

| Datasets | | GPin-TSVM<br>MCC<br>Time (s) | Lap-TSVM<br>MCC<br>Time (s) | Lap-PTSVM<br>MCC<br>Time (s) | LapGPin-TSVM<br>MCC<br>Time (s) |
|---|---|---|---|---|---|
| Fertility | Linear | 50.00±15.17 | 78.00±12.08 | 85.00±8.37 | **89.00±4.90** |
| | | 0.0140 | 0.0926 | 0.0453 | **0.1092** |
| | | 0.0256 | 0.0926 | 0.0366 | 0.0274 |
| | RBF | **88.00±4.00** | 87.00±4.00 | 86.00±5.83 | **88.00±4.00** |
| | | 0.0000 | -0.0153 | **0.0677** | 0.000 |
| | | 0.0268 | 0.0346 | 0.0364 | 0.0320 |
| Banknote | Linear | 93.88±2.36 | 97.45±0.8 | 97.38±1.16 | **98.54±0.73** |
| | | 0.8803 | 0.9485 | 0.9487 | **0.9704** |
| | | 0.2019 | 0.1419 | 0.1605 | 0.1552 |
| | RBF | 97.38±1.95 | 98.32±1.18 | 98.69±1.56 | **98.84±1.47** |
| | | 0.9473 | 0.9661 | 0.9739 | **0.9768** |
| | | 0.1054 | 0.1865 | 0.2345 | 0.2183 |
| Bupa | Linear | 63.77±3.89 | 63.76±6.01 | **68.12±4.10** | 67.25±2.98 |
| | | 0.2306 | 0.2474 | **0.3315** | 0.3129 |
| | | 0.0274 | 0.0403 | 0.0463 | 0.0411 |
| | RBF | 62.61±4.79 | 62.90±4.16 | 65.80±4.20 | **66.09±4.81** |
| | | 0.2228 | 0.2718 | 0.3221 | **0.3300** |
| | | 0.0270 | 0.0517 | 0.0495 | 0.0514 |
| Ionosphere | Linear | 78.89±4.25 | 83.18±1.95 | 82.89±4.38 | **84.32±5.12** |
| | | 0.5896 | 0.6360 | 0.6283 | **0.6576** |
| | | 0.0334 | 0.0506 | 0.0500 | 0.0377 |
| | RBF | 84.60±2.95 | 86.34±3.58 | 84.62±2.91 | **88.05±3.61** |
| | | 0.6584 | 0.7010 | 0.6611 | **0.7438** |
| | | 0.0378 | 0.0490 | 0.0512 | 0.0492 |
| Monk-2 | Linear | 73.40±3.79 | 69.93±6.92 | 77.57±5.45 | **79.17±1.53** |
| | | 0.4962 | 0.3903 | 0.5516 | **0.5878** |
| | | 0.0377 | 0.0385 | 0.0560 | 0.0480 |
| | RBF | 92.60±2.14 | 94.90±3.34 | 94.90±3.34 | **97.21±3.86** |
| | | 0.8542 | 0.8986 | 0.8993 | **0.9470** |
| | | 0.0362 | 0.0502 | 0.0541 | 0.0591 |
| Pima | Linear | 68.36±2.00 | 74.74±3.09 | 75.13±3.19 | **76.43±1.27** |
| | | 0.2287 | 0.4141 | 0.4251 | **0.4556** |
| | | 0.0534 | 0.0570 | 0.0791 | 0.0822 |
| | RBF | 73.45±4.57 | 75.27±3.04 | 75.14±3.08 | **76.18±3.03** |
| | | 0.3814 | 0.4227 | 0.4237 | **0.4487** |
| | | 0.0553 | 0.0840 | 0.0970 | 0.0982 |

## Table 16 (Cont.)

| | | | | | |
|---|---|---|---|---|---|
| Sonar | Linear | 65.38±6.91 | **68.79±4.80** | 67.31±5.41 | 67.31±5.58 |
| | | 0.3085 | **0.3804** | 0.3613 | 0.3613 |
| | | 0.0313 | 0.0463 | 0.0401 | 0.0396 |
| | RBF | 61.56±4.83 | 66.89±6.01 | 61.61±7.61 | **68.39±10.48** |
| | | 0.2221 | 0.3369 | 0.2459 | **0.3813** |
| | | 0.0301 | 0.0522 | 0.0418 | 0.0399 |
| Diabetes | Linear | 73.18±3.03 | 74.74±1.88 | **77.47±2.63** | 77.21±2.76 |
| | | 0.3766 | 0.4123 | **0.4829** | 0.4801 |
| | | 0.0540 | 0.0817 | 0.0895 | 0.0881 |
| | RBF | 74.22±1.74 | 75.40±3.41 | 74.48±2.61 | **75.66±3.61** |
| | | 0.4023 | 0.4462 | 0.4327 | **0.4449** |
| | | 0.0606 | 0.0843 | 0.1097 | 0.0968 |
| Spambase | Linear | 85.25±1.74 | 88.54±1.58 | 91.16±1.35 | **91.28±1.21** |
| | | 0.7233 | 0.7598 | 0.8142 | **0.8165** |
| | | 1.1479 | 1.2968 | 2.3394 | 2.1206 |
| | RBF | 86.12±1.70 | 88.17±1.46 | 90.38±1.23 | **90.58±1.07** |
| | | 0.7104 | 0.7510 | 0.7980 | **0.8020** |
| | | 1.3286 | 1.3474 | 2.7166 | 2.6171 |
| WDBC | Linear | 81.55±3.44 | 83.13±7.10 | 93.32±1.07 | **94.02±1.71** |
| | | 0.6239 | 0.6379 | 0.8574 | **0.8727** |
| | | 0.0446 | 0.0541 | 0.0696 | 0.0700 |
| | RBF | 91.73±3.27 | 92.80±2.67 | 91.57±2.10 | **92.97±2.47** |
| | | 0.8373 | 0.8495 | 0.8221 | **0.8501** |
| | | 0.0595 | 0.0636 | 0.0850 | 0.0752 |
| Australian | linear | 63.91±1.41 | 84.20±3.34 | 85.22±4.04 | **85.80±3.54** |
| | | 0.3266 | 0.6861 | 0.7056 | **0.7132** |
| | | 0.0516 | 0.0556 | 0.0841 | 0.0782 |
| | RBF | 64.05±3.02 | **71.45±2.65** | 68.41±3.85 | 70.00±3.32 |
| | | 0.2872 | **0.4249** | 0.3608 | 0.3944 |
| | | 0.0514 | 0.0701 | 0.0926 | 0.0932 |
| Heart | linear | 62.22±10.70 | 75.19±7.64 | 73.33±8.96 | **76.30±6.13** |
| | | 0.2552 | 0.5033 | 0.4636 | **0.5198** |
| | | 0.0341 | 0.0432 | 0.0458 | 0.0492 |
| | RBF | 70.37±5.61 | 78.15±1.38 | 80.00±2.15 | **80.74±2.77** |
| | | 0.4747 | 0.5570 | 0.5981 | **0.6141** |
| | | 0.0458 | 0.0553 | 0.0569 | 0.0553 |
| Spect heart | linear | 62.94±7.16 | 65.51±5.16 | 77.15±4.36 | **79.04±4.45** |
| | | **0.1669** | 0.1130 | 0.0525 | 0.0792 |
| | | 0.0373 | 0.0412 | 0.0503 | 0.0518 |
| | RBF | 64.88±10.93 | 58.43±6.85 | **70.04±2.59** | 69.66±3.21 |
| | | 0.1066 | 0.1136 | **0.2132** | 0.2100 |
| | | 0.0290 | 0.0411 | 0.0455 | 0.0384 |
| win/tile/loss | | 25/1/0 | 24/0/2 | 23/0/3 | |

### 3.2.4 Variation in Ratio of Unlabeled Data

LapGPin-TSVM extends GPin-TSVM by considering the incorporation of Laplacian techniques and transforming the model into a semi-supervised model. To test the performance of our model, we systematically changed the proportion of unlabeled data in the dataset. It considers ratios ranging from 20% to 80% of total data. The accuracy results of the linear and non-linear cases (RBF kernel) are illustrated in Table 13, 14, 15 and 16 which is for 20%, 40%, 60% and 80% of unlabeled data, respectively. We show the count of win/tile/loss of accuracy into the last column of each table. The average rank of accuracy and MCC score of all cases is computed and shown in Table 17. Here, ranks are assigned by ordering the values in ascending order, then the smallest values as rank 1. So, A higher average rank generally indicates better performance.

It can be observed in Tables 13, 14, 15 and 16 that in almost every case where the percentage of unlabeled data is increased, the GPin-TSVM method exhibits lower accuracy compared to other methods. The MCC score for this method consistently yield results in the same direction. This is because this method is a supervised learning approach that builds a classifier using only labeled data. Consequently, its performance is compromised when dealing with datasets that have a large amount of unlabeled data but limited labeled data.

As shown in Table 13, it is evident that our method outperforms others in both linear and non-linear cases on WDBC, Heart, and Specf heart data. However, for Sonar,Bupa and Monk-2 data, our method exhibits lower accuracy compared to Lap-TSVM and Lap-PTSVM. When considering the scenario where the proportion of unlabeled data is 40% of the total data in Table 14, it is observed that our method achieves the highest accuracy in Pima, Diabetes, and WDBC data. Similar to Table 15, our method continues to demonstrate the highest accuracy in both cases when considering Pima, Diabetes, Ionosphere and Monk-2 data. However, for Bupa and Specfheart data, Lap-PTSVM achieves the highest accuracy. Additionally, in the case of Sonar data, Lap-TSVM outperforms our approach in both cases. For Table 16, LapGPin-TSVM outperforms most other methods, except for the Bupa, Sonar , Diabetes, Australian and Specf heart data.

**Table 17 Average rank of all considering in different ratio of labeled data.**

| Model | Mean rank of Accuracy | | Mean rank of MCC | |
|---|---|---|---|---|
| | Linear | Nonlinear | Linear | Nonlinear |
| GPin-TSVM | 1.44 | 1.37 | 1.31 | 1.30 |
| Lap-TSVM | 2.38 | 2.43 | 2.44 | 2.42 |
| Lap-PTSVM | 3.05 | 2.85 | 2.84 | 2.89 |
| LapGPin-TSVM | **3.45** | **3.38** | **3.28** | **3.38** |

In addition to the evaluation based on accuracy, we now turn our attention to the results concerning the MCC score. In Table 13, the maximum MCC score for the non-linear scenario in Monk-2 data is consistent across Lap-TSVM, Lap-PTSVM, and our method, standing at 0.9464 closely approaching 1. Furthermore, in Table 14, within the same datasets, our LapGPin-TSVM consistently outperforms in both cases but the highest accuracy of linear case belong to Lap-TSVM. Notably, as indicated in Table 15, Lap-TSVM outperforms other methods in the Sonar data. Nevertheless, our proposed method surpasses the others in the Ionosphere and Monk-2 data. These outcomes demonstrate a similar pattern to those observed in Table 16.

**Table 18 The results of the Wilcoxon signed-rank test analysis of the model when examining changes in the ratio of unlabeled data.**

| Compare with | Negative ranks | | | Positive ranks | | | Test statistics | |
|---|---|---|---|---|---|---|---|---|
| **LapGPin-TSVM** | n | Mean rank | Sum of ranks | n | Mean rank | Sum of ranks | Ties | $p$-value |
| **Linear case** | | | | | | | | |
| GPin-TSVM | 0 | NaN | 0.00 | 51 | 26.00 | 1,326.00 | 1 | $\leq 0.001*$ |
| Lap-TSVM | 11 | 13.73 | 151.00 | 39 | 28.82 | 1,124.00 | 2 | $\leq 0.001*$ |
| Lap-PTSVM | 15 | 18.53 | 278.00 | 35 | 28.49 | 997.00 | 2 | $\leq 0.001*$ |
| **Nonlinear case** | | | | | | | | |
| GPin-TSVM | 1 | 11.50 | 11.50 | 46 | 24.27 | 1,116.50 | 5 | $\leq 0.001*$ |
| Lap-TSVM | 11 | 22.50 | 247.50 | 36 | 24.46 | 880.50 | 5 | $\leq 0.001*$ |
| Lap-PTSVM | 13 | 30.35 | 394.50 | 34 | 21.57 | 733.50 | 5 | 0.073 |

* Indicates statistically significant change

In the conclusion, within the linear scenario, LapGPin-TSVM shows superior performance in both accuracy and MCC, as evidenced by the values in Table 17, where it holds an average rank of 3.45 for accuracy and 3.38 for MCC, while Lap-PTSVM achieves 3.05 and 2.85, respectively. A higher average rank generally indicates better performance. Similarly, in the non-linear case, our proposed exhibits the best performance, as highlighted by the average rank values for accuracy and MCC, which are 3.28 and 3.38, as presented in Table 17. This investigation ensures emphasis on the ability of LapGPin-TSVM to utilize unlabeled instances, thereby leading to enhanced generalization and adaptability of decision boundaries.

### 3.2.5 Variation in Ratio of Noise

Due to the fact that LapGPin-TSVM utilizes error measurement through the computation of generalized pinball loss, a characteristic that follows this type of loss is its ability to handle noise issues in the data. Therefore, to test the model performance regarding this feature, we conducted experiments on datasets with

noise, having a zero mean and different variances of 0, 0.05, and 0.1. Here, we denote the percentage of noise in the data as $r$. The outcomes are displayed in Tables 19 and 20. Similarly, we computed and presented the average rank for all cases in Table 21.

The results are categorized into linear and non-linear cases. For the linear case presented in the Table 19, our method achieved the best performance at 70% (21 out of 30 instances). In the Fertility data, the experimental results for each model were comparable. In the Pima, Sonar, and Spambase data, our proposed loss outperformed other models. Considering the case where $r = 0.05$ in the Sonar data, although Lap-TSVM had the highest accuracy, the highest MCC score was achieved by Lap-PTSVM.

In Table 20, which displays the results of the nonlinear case, our model achieves the highest accuracy in 23 out of 30 instances, equating to 76.67%. Similarly to the previous findings in the Fertility data, all models produced closely aligned results. In the Bupa data, considering $r = 0$ and 0.05, our model achieved the highest accuracy, but the highest MCC values belonged to Lap-TSVM and Lap-PTSVM, respectively. Similar trends emerge in the Ionosphere and Monk-2 data when considering $r = 0.1$. Analyzing the average rank values for accuracy and MCC in the provided Table 21, it is evident that our method attained the highest average rank. This denotes superior performance compared to other models, with the second-ranking model being Lap-PTSVM. This outcome is attributed to the fact that the generalized pinball loss is derived from the pinball loss, which possesses the capability to handle noise. Therefore, in addition to our model being a generalized version of Lap-PTSVM, it also exhibits better sensitivity to noise.

This comprehensively assesses the robustness and performance under different levels of noise. The Laplacian regularization term in LapGPin-TSVM improves model generalization by considering the local data structure, particularly beneficial for datasets with complex intrinsic geometry.

**Table 19 The mean and standard deviation of tests accuracy with various noise on the UCI dataset were calculated using a linear kernel.**

| Datasets | r | GPin-TSVM<br>Acc (%)<br>MCC<br>Time (s) | Lap-TSVM<br>Acc (%)<br>MCC<br>Time (s) | Lap-PTSVM<br>Acc (%)<br>MCC<br>Time (s) | LapGPin-TSVM<br>Acc (%)<br>MCC<br>Time (s) |
|---|---|---|---|---|---|
| Fertility | 0 | **88.00±2.45** | 87.00±2.45 | **88.00±2.45** | **88.00±2.45** |
| | | 0.0000 | -0.0153 | **0.0939** | 0.000 |
| | | 0.0242 | 0.0297 | 0.0361 | 0.0292 |
| | 0.05 | **88.00±2.45** | **88.00±2.45** | **88.00±2.45** | **88.00±2.45** |
| | | 0.0000 | 0.0000 | 0.000 | 0.000 |
| | | 0.0233 | 0.0325 | 0.0220 | 0.0352 |
| | 0.1 | **88.00±2.45** | **88.00±2.45** | **88.00±2.45** | **88.00±2.45** |
| | | 0.0000 | 0.0000 | **0.0658** | 0.000 |
| | | 0.0206 | 0.0280 | 0.0311 | 0.0303 |
| Banknote | 0 | 94.39±1.97 | 98.24±1.09 | 98.25±0.70 | **98.40±1.04** |
| | | 0.8916 | 0.9654 | 0.9650 | **0.9675** |
| | | 0.1121 | 0.1076 | 0.1505 | 0.1667 |
| | 0.05 | 94.32±1.95 | 98.24±1.90 | 98.47±0.70 | **98.54±1.03** |
| | | 0.8903 | 0.9654 | 0.9694 | **0.9706** |
| | | 0.1027 | 0.1073 | 0.1612 | 0.1706 |
| | 0.1 | 94.46±2.24 | 98.25±1.09 | 98.32±0.67 | **98.54±0.46** |
| | | 0.8933 | 0.9654 | 0.9664 | **0.9705** |
| | | 0.111 | 0.1230 | 0.1597 | 0.1648 |
| Bupa | 0 | 63.48 ± 3.23 | 66.09±4.36 | 67.25 ± 4.82 | **67.54±4.45** |
| | | 0.2176 | 0.2774 | 0.3014 | **0.3193** |
| | | 0.0346 | 0.0365 | 0.0509 | 0.0433 |
| | 0.05 | 64.35±2.68 | 66.09±3.50 | 67.54±4.73 | **68.41±4.88** |
| | | 0.2387 | 0.2796 | 0.3080 | **0.3409** |
| | | 0.0260 | 0.0372 | 0.0505 | 0.0439 |
| | 0.1 | 64.05±2.65 | 66.09±3.95 | 66.67±4.93 | **68.12±4.39** |
| | | 0.2322 | 0.2779 | 0.2896 | **0.3341** |
| | | 0.0531 | 0.0406 | 0.0516 | 0.0424 |

**Table 19 (Cont.)**

| | | | | | |
|---|---|---|---|---|---|
| Ionosphere | 0 | 84.03±8.07 | **86.03±2.34** | 85.74±5.29 | 83.75±2.82 |
| | | 0.6499 | **0.7039** | 0.6901 | 0.6510 |
| | | 0.0352 | 0.0402 | 0.0546 | 0.0515 |
| | 0.05 | 81.18±4.77 | 84.33±0.91 | 84.33±1.60 | **85.76±1.23** |
| | | 0.5865 | 0.6603 | 0.6608 | **0.6962** |
| | | 0.0381 | 0.0494 | 0.0510 | 0.0540 |
| | 0.1 | 79.49±4.99 | 84.33±1.80 | 84.33±1.60 | **85.75±1.59** |
| | | 0.5515 | 0.6542 | 0.6612 | **0.6909** |
| | | 0.0383 | 0.0404 | 0.0520 | 0.0437 |
| Monk-2 | 0 | 79.19±6.15 | 78.95±4.38 | 78.50±6.45 | **80.11±5.23** |
| | | 0.5845 | 0.5771 | 0.5712 | **0.5987** |
| | | 0.0413 | 0.0507 | 0.0828 | 0.0962 |
| | 0.05 | 78.96±5.39 | 79.18±4.71 | 79.43±6.36 | **80.34±4.93** |
| | | 0.5788 | 0.5798 | 0.5894 | **0.6044** |
| | | 0.0443 | 0.0473 | 0.0880 | 0.0663 |
| | 0.1 | 78.26±6.22 | 78.72±4.55 | 78.96±6.41 | **79.41±5.01** |
| | | 0.5613 | 0.5702 | 0.5812 | **0.5860** |
| | | 0.0347 | 0.0503 | 0.0566 | 0.0705 |
| Pima | 0 | 75.78±1.29 | 77.60±1.48 | **77.73±0.77** | 77.60±1.58 |
| | | 0.4385 | 0.4821 | **0.4867** | 0.4843 |
| | | 0.0699 | 0.0796 | 0.0970 | 0.0863 |
| | 0.05 | 76.30±1.48 | 77.99±1.78 | **78.38±1.27** | 77.60±1.96 |
| | | 0.4535 | 0.4923 | **0.5026** | 0.4831 |
| | | 0.0624 | 0.0671 | 0.0908 | 0.0920 |
| | 0.1 | 76.18±1.92 | 76.82±1.46 | 77.73±1.28 | **78.00±1.11** |
| | | 0.4501 | 0.4644 | 0.4844 | **0.4929** |
| | | 0.0609 | 0.0679 | 0.1022 | 0.0910 |
| Sonar | 0 | 66.40±5.56 | 70.66±2.92 | **74.02±7.29** | 73.05±4.79 |
| | | 0.3387 | 0.4221 | **0.4811** | 0.4678 |
| | | 0.0340 | 0.0461 | 0.0429 | 0.0419 |
| | 0.05 | 64.38±7.85 | **71.14±4.61** | 71.13±5.66 | 69.71±3.27 |
| | | 0.3085 | 0.4247 | **0.4258** | 0.4043 |
| | | 0.0267 | 0.0424 | 0.0385 | 0.0404 |
| | 0.1 | 62.49±6.59 | 68.75±3.03 | 68.77±2.36 | **70.21±4.79** |
| | | 0.2653 | 0.3741 | 0.3811 | **0.4041** |
| | | 0.0278 | 0.0421 | 0.0430 | 0.0332 |

**Table 19 (Cont.)**

| | | | | | |
|---|---|---|---|---|---|
| Diabetes | 0 | 75.27±4.29 | 76.30±4.66 | 77.22±2.34 | **77.48±2.96** |
| | | 0.4240 | 0.4520 | 0.4770 | **0.4849** |
| | | 0.0902 | 0.0693 | 0.0971 | 0.0985 |
| | 0.05 | 76.05±4.03 | 76.56±4.60 | 77.48±3.07 | **78.13±2.87** |
| | | 0.4444 | 0.4584 | 0.4817 | **0.4980** |
| | | 0.0681 | 0.0570 | 0.1013 | 0.1179 |
| | 0.1 | 76.31±4.53 | 76.43±4.06 | 76.83±3.46 | **77.48±2.88** |
| | | 0.4534 | 0.4528 | 0.4679 | **0.4832** |
| | | 0.111 | 0.1230 | 0.1597 | 0.1648 |
| Spambase | 0 | 90.36±1.55 | **91.58±1.07** | 90.67±1.08 | 91.39±1.32 |
| | | 0.8019 | **0.8231** | 0.8039 | 0.8188 |
| | | 2.2855 | 1.4991 | 2.6345 | 3.1632 |
| | 0.05 | 89.19±1.93 | **91.34±0.95** | 90.82±1.10 | 90.47±1.15 |
| | | 0.7794 | **0.8181** | 0.8073 | 0.7999 |
| | | 2.3051 | 1.4790 | 2.632 | 2.9123 |
| | 0.1 | 88.97±1.51 | 90.30±0.70 | **90.47±0.52** | 89.75±0.86 |
| | | 0.7729 | 0.7961 | **0.8000** | 0.7848 |
| | | 2.2058 | 1.4714 | 2.7201 | 2.8718 |
| WDBC | 0 | 85.77±3.43 | 95.78±1.29 | **95.95±1.21** | 94.37±1.82 |
| | | 0.7086 | 0.9092 | **0.9135** | 0.8799 |
| | | 0.0586 | 0.0684 | 0.0789 | 0.0738 |
| | 0.05 | 83.13±1.48 | 93.15±1.87 | 94.02±2.58 | **94.03±1.15** |
| | | 0.6475 | 0.8525 | 0.8716 | **0.8729** |
| | | 0.0539 | 0.0698 | 0.0818 | 0.0731 |
| | 0.1 | 84.01±3.86 | 93.50±2.04 | 94.37±2.13 | **94.38±2.12** |
| | | 0.6686 | 0.8615 | **0.8804** | 0.8787 |
| | | 0.0609 | 0.0679 | 0.0984 | 0.1074 |
| win/tile/loss | | | 26/3/1 | 21/2/7 | 19/3/8 | |

**Table 20 The mean and standard deviation of tests accuracy with various noise on the UCI dataset were calculated using a RBF kernel.**

| Datasets | r | GPin<br>Acc (%)<br>MCC<br>Time (s) | Lap-Hinge<br>Acc (%)<br>MCC<br>Time (s) | Lap-Pin<br>Acc (%)<br>MCC<br>Time (s) | Lap-GP<br>Acc (%)<br>MCC<br>Time (s) |
|---|---|---|---|---|---|
| Fertility | 0 | **88.00±2.45** | 87.00±2.45 | 87.00±2.45 | **88.00±2.45** |
| | | 0.0000 | -0.0153 | **0.1593** | 0.000 |
| | | 0.0243 | 0.0331 | 0.0327 | 0.0334 |
| | 0.05 | **88.00±2.45** | **88.00±2.45** | **88.00±2.45** | **88.00±2.45** |
| | | 0.0000 | 0.0000 | -0.0306 | **0.1542** |
| | | 0.0239 | 0.0323 | 0.0324 | 0.0421 |
| | 0.1 | **88.00±2.45** | **88.00±2.45** | 87.00±2.45 | **88.00±2.45** |
| | | 0.0000 | 0.0000 | **0.0501** | 0.000 |
| | | 0.0232 | 0.0332 | 0.0332 | 0.0340 |
| Banknote | 0 | 98.90±0.89 | 99.48±0.72 | 99.34±0.58 | **100.00±0.00** |
| | | 0.9780 | 0.9898 | 0.9868 | **1.000** |
| | | 0.1315 | 0.1907 | 0.2853 | 0.2806 |
| | 0.05 | 98.90±0.89 | 99.34±0.99 | 99.42±0.55 | **100.00±0.00** |
| | | 0.9780 | 0.9870 | 0.9882 | **1.000** |
| | | 0.1447 | 0.2174 | 0.2851 | 0.2734 |
| | 0.1 | 98.76±1.14 | 99.34±0.81 | 99.49±0.50 | **100.00±0.00** |
| | | 0.9752 | 0.9869 | 0.9897 | **1.000** |
| | | 0.1388 | 0.2004 | 0.2735 | 0.2652 |
| Bupa | 0 | 68.11 ± 3.77 | 70.72±3.59 | 70.14 ± 5.31 | **70.72±6.37** |
| | | 0.3249 | **0.3917** | 0.3823 | 0.3749 |
| | | 0.0357 | 0.0387 | 0.0589 | 0.0580 |
| | 0.05 | 68.41±4.14 | 70.14±4.05 | 70.43±4.98 | **71.01±5.86** |
| | | 0.3313 | 0.3787 | **0.3861** | 0.3848 |
| | | 0.0404 | 0.0416 | 0.0571 | 0.0612 |
| | 0.1 | 68.41±4.61 | 70.43±3.12 | **70.72±5.05** | 70.43±4.72 |
| | | 0.3312 | 0.3857 | **0.3929** | 0.3773 |
| | | 0.0358 | 0.0464 | 0.0560 | 0.0624 |

**Table 20 (Cont.)**

| | | | | | |
|---|---|---|---|---|---|
| Ionosphere | 0 | 85.18±4.93 | **87.47±2.07** | 86.03±3.58 | 87.46±2.63 |
| | | 0.6781 | **0.7332** | 0.6990 | 0.7245 |
| | | 0.0412 | 0.0573 | 0.0525 | 0.0457 |
| | 0.05 | 84.03±6.50 | 86.32±1.96 | 87.75±1.13 | **88.88±3.45** |
| | | 0.6505 | 0.7077 | 0.7413 | **0.7613** |
| | | 0.0419 | 0.0502 | 0.0535 | 0.0481 |
| | 0.1 | 82.32±3.93 | 85.46±4.49 | 87.45±4.84 | **87.46±1.69** |
| | | 0.6174 | 0.6856 | **0.7305** | 0.7266 |
| | | 0.0382 | 0.0521 | 0.0478 | 0.0494 |
| Monk-2 | 0 | 94.20±4.53 | 96.05±3.93 | 96.29±3.41 | **96.99±3.34** |
| | | 0.8919 | 0.9216 | 0.9281 | **0.9420** |
| | | 0.0404 | 0.0579 | 0.0748 | 0.0747 |
| | 0.05 | 93.52±4.42 | 95.59±3.48 | 96.51±3.60 | **96.52±3.21** |
| | | 0.8799 | 0.9138 | 0.9323 | **0.9327** |
| | | 0.0550 | 0.0584 | 0.0700 | 0.0726 |
| | 0.1 | 91.44±3.34 | 95.36±2.45 | **95.36±2.66** | 94.21±3.75 |
| | | 0.8409 | **0.9073** | 0.9072 | 0.8912 |
| | | 0.0385 | 0.0676 | 0.0715 | 0.0718 |
| Pima | 0 | 76.04±2.99 | 76.17±2.65 | **77.21±1.71** | 76.56±2.60 |
| | | 0.4520 | 0.4545 | **0.4772** | 0.4624 |
| | | 0.0714 | 0.1229 | 0.1462 | 0.1324 |
| | 0.05 | 76.04±2.89 | 76.69±2.33 | **77.21±1.71** | 76.56±2.83 |
| | | 0.4526 | 0.4665 | **0.4776** | 0.4632 |
| | | 0.0779 | 0.1187 | 0.1450 | 0.1418 |
| | 0.1 | 76.04±2.71 | 76.30±2.71 | 76.43±1.52 | **76.69±2.83** |
| | | 0.4498 | **0.4756** | 0.4572 | 0.4662 |
| | | 0.0639 | 0.1137 | 0.1370 | 0.1359 |
| Sonar | 0 | 68.32±11.04 | 69.23±9.27 | 71.21±9.81 | **73.12±6.82** |
| | | 0.3671 | 0.3852 | 0.4232 | **0.4663** |
| | | 0.0296 | 0.0388 | 0.0427 | 0.0405 |
| | 0.05 | 69.71±3.23 | 72.13±2.32 | **73.03±7.23** | 71.64±2.77 |
| | | 0.3972 | 0.4496 | **0.4600** | 0.4400 |
| | | 0.0285 | 0.0425 | 0.0370 | 0.0431 |
| | 0.1 | 64.91±1.69 | 65.92±7.41 | 67.80±8.41 | **69.72±7.30** |
| | | 0.2931 | 0.3145 | 0.3564 | **0.3876** |
| | | 0.0295 | 0.0427 | 0.0418 | 0.0395 |

**Table 20 (Cont.)**

| | | | | | |
|---|---|---|---|---|---|
| Diabetes | 0 | 76.04±4.75 | 76.82±3.53 | 78.12±3.64 | **78.39±4.18** |
| | | 0.4445 | 0.4688 | 0.4939 | **0.5017** |
| | | 0.0645 | 0.1021 | 0.1357 | 0.1281 |
| | 0.05 | 76.56±4.45 | 76.69±3.33 | 77.87±4.10 | **78.39±4.18** |
| | | 0.4577 | 0.4632 | 0.4888 | **0.5015** |
| | | 0.0669 | 0.0927 | 0.1293 | 0.1216 |
| | 0.1 | 76.69±4.56 | 76.82±3.28 | 78.13±3.77 | **78.65±4.70** |
| | | 0.4593 | 0.4663 | 0.4958 | **0.5073** |
| | | 0.0703 | 0.0914 | 0.1279 | 0.1239 |
| Spambase | 0 | 89.01±1.38 | 90.04±1.66 | 90.75±1.29 | **90.91±1.15** |
| | | 0.7688 | 0.7905 | 0.8057 | **0.8089** |
| | | 2.8210 | 3.9467 | 4.2431 | 3.9955 |
| | 0.05 | 89.06±1.08 | 89.88±1.48 | 90.49±1.01 | **90.56±0.62** |
| | | 0.7699 | 0.7873 | 0.8003 | **0.8018** |
| | | 2.7608 | 4.0191 | 3.9964 | 3.7862 |
| | 0.1 | 88.16±0.93 | 89.42±1.30 | **89.58±0.99** | 89.43±0.82 |
| | | 0.7513 | 0.7780 | **0.7808** | 0.7776 |
| | | 2.6906 | 3.9297 | 3.9392 | 3.6749 |
| WDBC | 0 | 92.99±0.96 | 93.32±1.22 | 93.84±2.31 | **94.72±1.78** |
| | | 0.8499 | 0.8574 | 0.8680 | **0.8869** |
| | | 0.0588 | 0.0681 | 0.0893 | 0.0878 |
| | 0.05 | 92.97±0.96 | 93.49±1.34 | 94.02±2.34 | **94.99±1.72** |
| | | 0.8497 | 0.8614 | 0.8719 | **0.8907** |
| | | 0.0544 | 0.0736 | 0.0817 | 0.0930 |
| | 0.1 | 93.33±1.62 | 93.66±1.32 | 94.02±2.27 | **94.72±1.26** |
| | | 0.8572 | 0.8655 | 0.8722 | **0.8871** |
| | | 0.0605 | 0.0698 | 0.0941 | 0.0885 |
| win/tile/loss | | | 27/3/0 | 23/3/4 | 23/1/6 | |

**Table 21 Average rank of all considering in different ratio of noise.**

| Model | Mean rank of Accuracy | | Mean rank of MCC | |
|---|---|---|---|---|
| | Linear | Nonlinear | Linear | Nonlinear |
| GPin-TSVM | 1.27 | 1.20 | 1.23 | 1.13 |
| Lap-TSVM | 2.40 | 2.28 | 2.32 | 2.45 |
| Lap-PTSVM | 3.00 | 2.97 | 3.15 | 3.10 |
| LapGPin-TSVM | **3.33** | **3.55** | **3.30** | **3.32** |

Furthermore, we have also conducted a statistical analysis to assess the differences between the model we propose and other models. Due to the non-normal distribution of our data [34], we chose to employ the Wilcoxon signed-rank test [37]. This test is a non-parametric statistical test employed to compare two related samples and determine whether there is a significant difference between the paired observations in a sample, employing a significance level of 0.05 for our analysis.

In this analysis, we employ accuracy as the metric, and the results are presented in Tables 18 and 22. Table 18 compares our model with others, incorporating data from Tables 13, 14, 15, and 16, which present results based on the ratio of unlabeled samples. Table 22, on the other hand, extracts data from Tables 19 and 20. As observed in both tables, LapGPin-TSVM exhibits significant differences compared to GPin-TSVM, Lap-TSVM, and Lap-PTSVM.

**Table 22 The results of the Wilcoxon signed-rank test analysis of the model when examining changes in the ratio of noise.**

| Compare with | Negative ranks | | | Positive ranks | | | Test statistics | |
|---|---|---|---|---|---|---|---|---|
| **LapGPin-TSVM** | n | Mean rank | Sum of ranks | n | Mean rank | Sum of ranks | Ties | $p$-value |
| **Linear case** | | | | | | | | |
| GPin-TSVM | 1 | 1.00 | 1.00 | 26 | 14.50 | 377.00 | 3 | $\leq 0.001$* |
| Lap-TSVM | 7 | 12.86 | 90.00 | 21 | 15.05 | 361.00 | 2 | 0.010* |
| Lap-PTSVM | 8 | 17.06 | 136.50 | 19 | 12.71 | 241.50 | 3 | 0.207 |
| **Nonlinear case** | | | | | | | | |
| GPin-TSVM | 0 | NaN | 0.00 | 27 | 14.00 | 378.00 | 3 | $\leq 0.001$* |
| Lap-TSVM | 4 | 7.88 | 31.50 | 23 | 15.07 | 346.50 | 3 | $\leq$0.001* |
| Lap-PTSVM | 6 | 15.67 | 94.00 | 23 | 14.83 | 341.00 | 1 | 0.008* |

* Indicates statistically significant change

# CHAPTER IV

# UNCERTAIN DATA CLASSIFICATION USING SUPPORT VECTOR MACHINE WITH GENERALIZED PINBALL LOSS

## 4.1 Support vector machine with generalized pinball loss function for uncertain data classification (UGPinSVM)

Consider uncertainty of the training samples, we give a multivariate Gaussian distribution with mean vector $\mathbf{x}_i \in \mathbb{R}^n$ and covariance matrix $\Sigma_i \in \mathbb{R}^{n\times n}$ $(i = 1, \ldots, m)$ as $(\mathbf{x}_1, \Sigma_1, y_1), (\mathbf{x}_2, \Sigma_2, y_2), \ldots, (\mathbf{x}_m, \Sigma_m, y_m)$, where $y_i \in \{1, -1\}$.

First of all, the idea of introducing uncertain data will be discussed. The original features of samples are regarded as the means of uncertain samples. The covariance matrix of the $i$th sample in class $k$ is defined by the following diagonal matrices,

$$S_i^k = \begin{bmatrix} 0.25c_i^k r_1 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & 0.25c_i^k r_n \end{bmatrix}$$

where $r_j (j = 1, \ldots, n)$ is the range of $i$th feature of the original features and $c_i^k$ is the uniform distribution between the interval $(0.95\bar{c}_k, 1.05\bar{c}_k)$. Different $\bar{c}_k$ values are to make different classes have different levels of uncertainty.

Now, we will provide the Python code for constructing the covariance matrix, generating uncertain data, and explaining the method we use to obtain solutions for problems, all while describing the content in detail.

To execute the code for this work, we need to import the following.

```
import numpy as np
import math
from scipy.integrate import quad
from scipy import integrate
import random
from sklearn.base import BaseEstimator
```

In order to obtain the covariance of a multivariate normal distribution, we need to construct the range of the $i$th feature using the sigma function. Subsequently, this range becomes the diagonal matrix, where each element corresponds to a feature's range. It is important to note that all the code presented here is extracted from the Class(BaseEstimator), and as a result, the input self is utilized within the function. If you intend to use the code, kindly review it once more for clarity and accuracy.

```
def sigma(self,x):
    d = []
    # i run following the number of data features
    for i in range(len(x[0])):
        m = max(x[:,i])
        n = min(x[:,i])
        v = m - n
        d.append(v)
    sig = np.diag(d)
    return sig
```

Next, we compute the value of $c_i^k$. In this work, which involves two classes of data, we assume that $\bar{c}_1 = 0.05$ for the positive class and $\bar{c}_2 = 0.07$ for the negative class.

```
def kvtest(self,x,t):
    v = []
    for n in range(len(x)):
        if t[n] == 1:
            #for positive class c_1=0.05
            k_0 = random.uniform(0.95*0.05,1.05*0.05)
        elif t[n] == -1:
            #for negative class c_1=0.07
            k_0 = random.uniform(0.95*0.07,1.05*0.07)
        v.append(k_0)
    return v
```

After obtaining the covariance and the original features of samples serve as the means, we utilize these statistical parameters to construct new data points. This process captures uncertainty through the perspective of a multivariate normal distribution.

```
def construct_x_uncertain(self,x,t):
    x_new = []
    kv = self.kvtest(x,t)
    sigma = self.sigma(x)
    for i in range(len(x)):
        sigma_new = 0.25*kv[i]*sigma
        cov = sigma_new
        mean = x[i]
        x_uncertain = np.random.multivariate_normal(mean, cov)
        x_new.append(x_uncertain)
    return np.array(x_new)
```

The multivariate normal distribution allows us to characterize the relationships and dependencies among different features, providing a comprehensive understanding of the uncertainty inherent in the generated data points.

To improve the performance of UPinSVMs, we propose a new model that incorporates the concept of the generalized pinball loss function for uncertain data classification. The formulation of our problems is as follows:

$$\min_{\mathbf{w},b} \frac{1}{2}(\|\mathbf{w}\|^2 + b^2) + C\sum_{i=1}^{m}\int_{\mathbb{R}^n} \mathcal{L}_{\tau_1,\tau_2}^{\epsilon_1,\epsilon_2}\big(1 - y_i(\mathbf{w}^\top\mathbf{x} + b)\big) f_{X_i}(\mathbf{x})d\mathbf{x}. \tag{4.1.1}$$

To handle (4.1.1), we define two halfspaces for the uncertain sample $i$: $\Omega_i^1 = \big\{\mathbf{x} : \tau_1\big(1 - y_i(\mathbf{w}^\top\mathbf{x} + b)\big) \geq \epsilon_1\big\}$ and $\Omega_i^2 = \big\{\mathbf{x} : -\tau_2\big(1 - y_i(\mathbf{w}^\top\mathbf{x} + b)\big) \geq \epsilon_2\big\}$. Then, we determine the loss function of the $i$th uncertain samples based on those two halfspaces as follows.

$$\begin{aligned} L_i(\mathbf{w}, b) = &\int_{\Omega_i^1} \Big(\tau_1\big(1 - y_i(\mathbf{w}^\top\mathbf{x} + b)\big) - \epsilon_1\Big) f_{X_i}(\mathbf{x})d\mathbf{x} \\ &+ \int_{\Omega_i^2} \Big(-\tau_2\big(1 - y_i(\mathbf{w}^\top\mathbf{x} + b)\big) - \epsilon_2\Big) f_{X_i}(\mathbf{x})d\mathbf{x}. \end{aligned} \tag{4.1.2}$$

Utilizing random vector $X_i$, we define two random variables: $\bar{X}_i = \tau_1\big(1 - y_i(\mathbf{w}^\top X_i + b)\big) - \epsilon_1$ and $\hat{X}_i = -\tau_2\big(1 - y_i(\mathbf{w}^\top X_i + b)\big) - \epsilon_2$. Given that $X_i$ is a random variable with a Gaussian distribution with a mean of $\mathbf{x}_i$ and a covariance of $\Sigma_i$, the variables $\bar{X}_i$ and $\hat{X}_i$ are both Gaussian random variables. From the work of Liang [23], we obtain the mean and variance of $\bar{X}_i$ are $\tau_1\big(1 - y_i(\mathbf{w}^\top\mathbf{x}_i + b)\big) - \epsilon_1$ and $\mathbf{w}^\top\Sigma_i\mathbf{w}$, respectively, and the mean and variance of $\hat{X}_i$ are $-\tau_2\big(1 - y_i(\mathbf{w}^\top\mathbf{x}_i + b)\big) - \epsilon_2$

and $\mathbf{w}^\top \Sigma_i \mathbf{w}$, respectively. The random vector $X_i$ can be replaced by the random variables $\bar{X}_i$ and $\hat{X}_i$ to represent $L_i(\mathbf{w}, b)$ in (4.1.2) as follow:

$$L_i(\mathbf{w}, b) = \int_0^\infty \mathbf{z} f_{\bar{X}_i}(\mathbf{z}) d\mathbf{z} + \int_0^\infty \mathbf{z} f_{\hat{X}_i}(\mathbf{z}) d\mathbf{z}, \tag{4.1.3}$$

where $f_{\bar{X}_i}(\mathbf{z})$ and $f_{\hat{X}_i}(\mathbf{z})$ are the probability density functions of $\bar{X}_i$ and $\hat{X}_i$, respectively. Thus, the high-dimensional integral problem becomes the one-dimensional integral problem. From Section 2.2, using Lemma 2.8.8, $L_i(\mathbf{w}, b)$ in (4.1.3) can be written in the term of the complementary error function.

$$\begin{aligned} L_i(\mathbf{w}, b) =& \frac{d_i^1}{2}\left(1 - \operatorname{erf}\left(f_i^1\right)\right) + \frac{e_i}{2\sqrt{\pi}} \exp\left(-\left(f_i^1\right)^2\right) \\ &+ \frac{d_i^2}{2}\left(1 - \operatorname{erf}\left(f_i^2\right)\right) + \frac{e_i}{2\sqrt{\pi}} \exp\left(-\left(f_i^2\right)^2\right), \end{aligned} \tag{4.1.4}$$

where $d_i^1 = \tau_1\left(1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)\right) - \epsilon_1, d_i^2 = -\tau_2\left(1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)\right) - \epsilon_2, e_i = \sqrt{2\mathbf{w}^\top \Sigma_i \mathbf{w}}, f_i^1 = -\frac{d_i^1}{e_i}$, and $f_i^2 = -\frac{d_i^2}{e_i}$. We use (4.1.4) to the problem (4.1.1),

$$\min_{\mathbf{w},b} h(\mathbf{w}, b) = \frac{1}{2}(\|\mathbf{w}\|^2 + b^2) + C \sum_{i=1}^{m} L_i(\mathbf{w}, b). \tag{4.1.5}$$

There are no constraints on $b$ and $\mathbf{w}$ in the proposed model. The model (4.1.5) can be solved using various gradient descent techniques, but first, we need to determine the gradient of the function. In (4.1.5), the gradient of the function $L_i(\mathbf{w}, b)$ with respect to $\mathbf{w}$ is given by the following expression:

$$\begin{aligned} \frac{\partial L_i(\mathbf{w}, b)}{\partial \mathbf{w}} =& \frac{\tau_2\left(1 - \operatorname{erf}\left(f_i^2\right)\right) - \tau_1\left(1 - \operatorname{erf}\left(f_i^1\right)\right)}{2} y_i \mathbf{x}_i \\ &+ \frac{\exp\left(-\left(f_i^2\right)^2\right) + \exp\left(-\left(f_i^1\right)^2\right)}{e_i\sqrt{\pi}} \Sigma_i \mathbf{w}. \end{aligned} \tag{4.1.6}$$

The gradient of the function $L_i(\mathbf{w}, b)$ with respect to $b$ is denoted by

$$\frac{\partial L_i(\mathbf{w}, b)}{\partial b} = \frac{\tau_2\left(1 - \operatorname{erf}\left(f_i^2\right)\right) - \tau_1\left(1 - \operatorname{erf}\left(f_i^1\right)\right)}{2} y_i. \tag{4.1.7}$$

Let us look at the python code for compute the gradient, which is shown below. For the input sigma and kv in the function gradient, we obtain from the previous code.

As the gradient of the function $L_i(\mathbf{w}, b)$ with respect to $\mathbf{w}$, we can write the code using function gradient.

```
def wgradient(self, w, b, x, t, sigma, kv):
    gradient = []
    for k in range(len(t)):
        #d_1
        dx_1 = self.tau_1*(1 - t[k]*(np.dot(w.T,x[k]) + b)) -
            self.epsilon_1
        #d_2
        dx_2 = - self.tau_2*(1 - t[k]*(np.dot(w.T,x[k]) + b)) -
             self.epsilon_2
        #covariance matrix
        sigma_new = 0.25*kv[k]*sigma
        #e_i
        dsigma =  math.sqrt(2*np.dot(np.dot(w.T,sigma_new),w))
        g1 = ((math.exp(-(dx_2**2)/(dsigma**2)) + math.exp(-(
            dx_1**2)/(dsigma**2)))/(math.sqrt(math.pi)*dsigma))
             * np.dot(sigma_new,w)

        f = lambda y : math.exp(-y**2)
        #erf(-dx_1/dsigma)
        integ_1 = integrate.quad(f, 0, (-dx_1/dsigma))
        #erf(-dx_2/dsigma)
        integ_2 = integrate.quad(f, 0, (-dx_2/dsigma))

        g2 =  ((self.tau_2*(1- (2/math.sqrt(math.pi))*integ_2
            [0]) - (self.tau_1*(1- (2/math.sqrt(math.pi))*
            integ_1[0])))/2)*x[k]*t[k]

        g = g1 + g2
        gradient.append(g)
    return wgradient
```

For the gradient of the function $L_i(\mathbf{w}, b)$ with respect to $b$, we using function bgradient.

```
def bgradient(self, w, b, x, t, sigma, kv):
    bgradient = []
    for k in range(len(t)):
        #d_1
        dx_1 = self.tau_1*(1 - t[k]*(np.dot(w.T,x[k]) + b)) -
            self.epsilon_1
```

```
            #d_2
            dx_2 = - self.tau_2*(1 - t[k]*(np.dot(w.T,x[k]) + b)) -
                self.epsilon_2
            #covariance matrix
            sigma_new = 0.25*kv[k]*sigma
            #e_i
            dsigma =  math.sqrt(2*np.dot(np.dot(w.T,sigma_new),w))

            #erf(-dx_1/dsigma)
            f = lambda y : math.exp(-y**2)
            integ_1 = integrate.quad(f, 0, (-(dx_1)/dsigma))
            #erf(-dx_2/dsigma)
            integ_2 = integrate.quad(f, 0, (-(dx_2)/dsigma))

            bg =  ((self.tau_2*(1- (2/math.sqrt(math.pi))*integ_2
                [0]) - (self.tau_1*(1- (2/math.sqrt(math.pi))*
                integ_1[0])))/2)*t[k]

            bgradient.append(bg)
        return bgradient
```

After obtaining the gradient of the function, we proceed to generate the code for the problem (4.1.5) as follows:

```
    def cost_function(self,w,b,x,t):
        loss = []
        #using function kvtest
        kv = self.kvtest(x,t)
        #using function sigma
        sigma = self.sigma(x)
        for k in range(len(t)):
            #d_1
            dx_1 = self.tau_1*(1 - t[k]*(np.dot(w.T,x[k]) + b)) - self.
                epsilon_1
            #d_2
            dx_2 = - self.tau_2*(1 - t[k]*(np.dot(w.T,x[k]) + b)) -
                self.epsilon_2
            #covariance matrix
            sigma_new = 0.25*kv[k]*sigma
            #e_i
```

```
15          dsigma =  math.sqrt(2*np.dot(np.dot(w.T,sigma_new),w))
16
17          f = lambda y : math.exp(-y**2)
18          #erf(-dx_1/dsigma)
19          integ_1 = integrate.quad(f, 0, (-dx_1/dsigma))
20          #erf(-dx_2/dsigma)
21          integ_2 = integrate.quad(f, 0, (-dx_2/dsigma))
22
23          g1 = ((dx_1)/2)*(1- (2/math.sqrt(math.pi))*integ_1[0]) +
                dsigma/(2*math.sqrt(math.pi))*(math.exp(-(dx_1**2/
                dsigma**2)))
24          g2 = ((dx_2)/2)*(1- (2/math.sqrt(math.pi))*integ_2[0]) +
                dsigma/(2*math.sqrt(math.pi))*(math.exp(-(dx_2**2/
                dsigma**2)))
25
26          loss_ = g1 + g2
27          loss.append(loss_)
28
29      loss = np.array(loss)
30      #form of the problem
31      cost = 1/2*np.linalg.norm(w)**2 +(1/2)*(b**2)+self.C*np.mean(
            loss)
32      return loss, cost
```

Now, we are in the process of exploring the properties of our proposed work. Let $\psi(\mathbf{w}, b) = \sum_{i=1}^{m} L_i(\mathbf{w}, b)$. Referring to (4.1.6) and (4.1.7), we have the following proposition.

**Proposition 4.1.1.** Assume that $\Sigma_1, \ldots, \Sigma_m$ are positive definite, i.e. $\lambda_{\min}(\Sigma_1) > 0, \ldots, \lambda_{\min}(\Sigma_m) > 0$, where $\lambda_{\min}(\Sigma_i)$ denotes the smallest eigenvalue of the matrix $\Sigma_i$. Then the norm of $\frac{\partial \psi(\mathbf{w},b)}{\partial b}$ is bounded above by the constant $U_b = m$ and the norm of $\frac{\partial \psi(\mathbf{w},b)}{\partial \mathbf{w}}$ is bounded above by the constant $U_{\mathbf{w}}$, denoted by

$$U_{\mathbf{w}} = \sum_{i=1}^{m} \left( \|\mathbf{x}_i\| + \frac{\|\Sigma_i\|_F}{\sqrt{2\pi\lambda_{\min}(\Sigma_i)}} \right). \tag{4.1.8}$$

*Proof.* The positive definiteness of the matrices $\Sigma_1, \ldots, \Sigma_m$ guarantrees that the smallest eigenvalue of each matrix is bigger than zero. Note that the function $\text{erf}(\mathbf{x})$ takes values in $[-1, 1]$ and $\exp\left( -\left(f_i^2\right)^2 \right)$ takes values in $[0, 1]$. We have

$\|e_i\| \geq \|\mathbf{w}\|\sqrt{2\lambda_{\min}(\Sigma_i)}$. Thus,

$$\begin{aligned}\left\|\frac{\partial\psi(\mathbf{w},b)}{\partial\mathbf{w}}\right\| &= \left\|\sum_{i=1}^{m}\left[\frac{\tau_2(1-\operatorname{erf}(f_i^2))-\tau_1(1-\operatorname{erf}(f_i^1))}{2}y_i\mathbf{x}_i\right.\right.\\ &\left.\left.+\frac{\exp\left(-\left(f_i^2\right)^2\right)+\exp\left(-\left(f_i^1\right)^2\right)}{e_i\sqrt{\pi}}\Sigma_i\mathbf{w}\right]\right\|\\ &\leq \sum_{i=1}^{m}\left\|\frac{\tau_2(1-\operatorname{erf}(f_i^2))-\tau_1(1-\operatorname{erf}(f_i^1))}{2}y_i\mathbf{x}_i\right\|\\ &+\sum_{i=1}^{m}\left\|\frac{\exp\left(-\left(f_i^2\right)^2\right)+\exp\left(-\left(f_i^1\right)^2\right)}{e_i\sqrt{\pi}}\Sigma_i\mathbf{w}\right\|\\ &\leq \sum_{i=1}^{m}\|\mathbf{x}_i\|+2\sum_{i=1}^{m}\frac{\|\Sigma_i\|_F}{\sqrt{2\pi\lambda_{\min}(\Sigma_i)}}.\end{aligned}$$

and

$$\left\|\frac{\partial\psi(\mathbf{w},b)}{\partial b}\right\| = \left\|\frac{\tau_2(1-\operatorname{erf}(f_i^2))-\tau_1(1-\operatorname{erf}(f_i^1))}{2}y_i\right\| \leq m.$$

□

This proposition demonstrates that the bounded of functions ensures that the error does not change violently. In this work, we employ the stochastic gradient descent solver to address the proposed optimization model.

To apply the stochastic gradient descent (SGD) algorithm, let $t > 0$ be a number of iteration. We assume that at time $t$, we are given a batch of $k$ training examples from two classes denoted by $\hat{\theta}_t = \{(\mathbf{x}_{t,1}, y_{t,1}), (\mathbf{x}_{t,2}, y_{t,2}), \ldots, (\mathbf{x}_{t,k}, y_{t,k})\}$. Let $\mathbf{w}_t$ represent the current iteration of the hyperplane. Therefore, the stochastic gradients of $h(\mathbf{w}_t, \hat{\theta}_t)$ associated with the samples $\hat{\theta}_t$ are calculated at the point $\mathbf{w}_t$. This can be expressed as follow:

$$\hat{s}_{\mathbf{w}}(\mathbf{w}, b; \hat{\theta}_t) = \frac{\partial h(\mathbf{w}_t, b; , \hat{\theta}_t)}{\partial \mathbf{w}} = \mathbf{w}_t + \frac{C}{k}\sum_{i=1}^{k}\frac{\partial L_i(\mathbf{w}_t, b; \hat{\theta}_t)}{\partial \mathbf{w}}, \tag{4.1.9}$$

$$\hat{s}_b(\mathbf{w}, b; \hat{\theta}_t) = \frac{\partial h(\mathbf{w}_t, b; , \hat{\theta}_t)}{\partial b} = b_t + \frac{1}{k}\sum_{i=1}^{k}\frac{\partial L_i(\mathbf{w}_t, b; \hat{\theta}_t)}{\partial b}. \tag{4.1.10}$$

Hence, update iteration $\mathbf{w}_t, b_t$ are as follow:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t\hat{s}_{\mathbf{w}}(\mathbf{w}, b; , \hat{\theta}_t), \tag{4.1.11}$$

$$b_{t+1} = b_t - \eta_t \hat{s}_b(\mathbf{w}, b; , \hat{\theta}_t), \tag{4.1.12}$$

where $\eta_t$ is the step size. The algorithmic steps for the procedure can be found in Algorithm 2.

**Algorithm 2** The SGD method to solve (4.1.5)

**Input:** Variable $\mathbf{w}_0$,

$k \in \{1, 2, \ldots, m\}$,

batch size of $k$ training examples denoted by $\hat{\theta}_t$,

positive parameters $C, \tau_1, \tau_2, \epsilon_1, \epsilon_2$ and

Maximum number of iteration $T$.

**Output:** Optimal hyperplane parameter $\mathbf{w}$

1: **for** $t = 0, 1, 2, \ldots T$ **do**

2: Acquire $\theta_t = \{(\mathbf{x}_t, y_t)\}$;

3: Compute stochastic gradient $\hat{s}_{\mathbf{w}}(\mathbf{w}, b; \hat{\theta}_t)$ using (4.1.9) and $\hat{s}_b(\mathbf{w}, b; \hat{\theta}_t)$ using (4.1.10) to update $\mathbf{w}_{t+1}$ and $b_{t+1}$;

4: Updating $\mathbf{w}_{t+1}$ using (4.1.11) and $b_{t+1}$ using (4.1.12);

5: **end for**

6: Set $\mathbf{w} = \mathbf{w}_{t+1}$ and $b = b_{t+1}$.

The Python code for this framework is described as follows. Note that we will combine all the above functions and proceed to run the process.

```
1       def fit(self, x, t):
2           #using function construct_x_uncertain to tranform exact
                vector into random vector
3           x_new = self.construct_x_uncertain(x,t)
4           #checks for labels
5           self.classes_ = np.unique(t)
6           #initail variables
7           it = 0
8           w = np.ones(len(x[0]))
9           b = 1
10          obj_func = []
11          obj_batchsgd = []
12          iter_batchsgd = []
13          for epoch in range(self.max_epochs):
```

```
14              idx = np.random.permutation(len(t))
15              print("Epoch: %d" %(epoch+1), idx)
16              for i in range(len(t)):
17                  r = idx[i*self.n_batches:(i+1)*self.n_batches]
18                  if r.size==0: break
19                  it = it + 1
20                  iter_batchsgd.append(it)
21                  #compute cost function
22                  loss, cost = self.cost_function(w,b,x,t)
23                  obj_func.append(cost)
24                  print("----Iteration: %d" %(i+1), r)
25                  X = x[r,:]
26                  T = t[r]
27                  sigma = self.sigma(X)
28                  kv = self.kvtest(X,T)
29                  X_new = x_new[r,:]
30                  T_new = t[r]
31                  #compute gradient of loss depend on w
32                  gradloss = self.wgradient(w,b,X_new,T_new,sigma, kv
                        )
33                  gradloss = np.vstack(gradloss)
34                  gloss = np.mean(gradloss, axis = 0)
35                  grad =  w + self.C*gloss
36                  #step size
37                  eta =1/it
38                  #update weight
39                  w -= eta*grad
40                  print("----w: ",w)
41                  #compute gradient of loss depend on b
42                  bgradloss1 = self.bgradient(w,b,X_new,T_new,sigma,
                        kv)
43                  bgradloss2 = np.vstack(bgradloss1)
44                  bgradloss = np.mean(bgradloss2)
45                  bgrad = b + self.C*bgradloss
46                  #update bias
47                  b -= eta*bgrad
48                  print("----b: ",b)
49          self.final_iter = it
50          self._coef = w
51          self._intercept = b
```

```
52          self.obj_func = obj_func
53          self.obj_batchsgd = obj_batchsgd
54          self.iter_batchsgd = iter_batchsgd
55          return self
```

In the next section, we examine the characteristics of the proposed model in terms of its ability to handle noise insensitivity and minimize scatter.

### 4.1.1 Noise insensitivity

Consider the generalized sign function $\operatorname{sgn}_{\tau_1,\tau_2}^{\epsilon_1,\epsilon_2}(v)$ defined by

$$\operatorname{sgn}_{\tau_1,\tau_2}^{\epsilon_1,\epsilon_2}(v) = \begin{cases} \{\tau_1\}, & v > \frac{\epsilon_1}{\tau_1}, \\ [0,\tau_1], & v = \frac{\epsilon_1}{\tau_1}, \\ \{0\}, & -\frac{\epsilon_2}{\tau_2} < v < \frac{\epsilon_1}{\tau_1}, \\ [-\tau_2,0], & v = -\frac{\epsilon_2}{\tau_2}, \\ \{-\tau_2\}, & v < -\frac{\epsilon_2}{\tau_2}. \end{cases} \tag{4.1.13}$$

Note that $\operatorname{sgn}_{\tau_1,\tau_2}^{\epsilon_1,\epsilon_2}(v)$ is the subgradient of the generalized pinball loss function defined by (2.6.1). According to the function $\operatorname{sgn}_{\tau_1,\tau_2}^{\epsilon_1,\epsilon_2}(v)$, we can express the necessary conditions of the optimization problem of (4.1.1) as follows:

$$\mathbf{0} = \mathbf{w} + C\sum_{i=1}^{m}\int_{\mathbb{R}^n} \operatorname{sgn}_{\tau_1,\tau_2}^{\epsilon_1,\epsilon_2}\big(1 - y_i(\mathbf{w}^\top\mathbf{x} + b)\big)(-y_i\mathbf{x}) f_{X_i}(\mathbf{x})d\mathbf{x}, \tag{4.1.14}$$

and

$$0 = b + C\sum_{i=1}^{m}\int_{\mathbb{R}^n} \operatorname{sgn}_{\tau_1,\tau_2}^{\epsilon_1,\epsilon_2}\big(1 - y_i(\mathbf{w}^\top\mathbf{x} + b)\big)(-y_i) f_{X_i}(\mathbf{x})d\mathbf{x}. \tag{4.1.15}$$

Considering the integral problems in (4.1.14) and (4.1.15), the boundary points from uncertain samples will not affect (4.1.14) and (4.1.15). According to the $i$th uncertain sample, we define the set $\Omega_i^3 = \big\{\mathbf{x} : \tau_1\big(1 - y_i(\mathbf{w}^\top\mathbf{x} + b)\big) \leq \epsilon_1, -\tau_2\big(1 - y_i(\mathbf{w}^\top\mathbf{x} + b)\big) \leq \epsilon_2\big\}$. Using $\Omega_i^1, \Omega_i^2$ and $\Omega_i^3$, we can represent Eq. (4.1.14) into the following form,

$$\mathbf{0} = \mathbf{x} + C\tau_1\sum_{i=1}^{m}\int_{\Omega_i^1}(-y_i\mathbf{x}) f_{X_i}(\mathbf{x})d\mathbf{x} - C\tau_2\sum_{i=1}^{m}\int_{\Omega_i^2}(-y_i\mathbf{x}) f_{X_i}(\mathbf{x})d\mathbf{x}. \tag{4.1.16}$$

Due to the existence of the integral problem, boundary points of the $i$th uncertain sample are combined into the sets $\Omega_i^1, \Omega_i^2$ and $\Omega_i^3$, as contrasted to deterministic samples where boundary points are considered independently. Eq. (4.1.16) shows that the parameters $\tau_1$ and $\tau_2$ limit the number of data points in the sets $\Omega_i^1$ and $\Omega_i^2$. If $\tau_1 = 1$ and $\tau_2 = 1$, both sets have many data points, making them less sensitive to feature noise. When $\tau_1$ and $\tau_2$ are small, there are only a few data points in the sets $\Omega_i^1$ and $\Omega_i^2$. If $\tau_1 = 0$ and $\tau_2 = 0$, we will disregard the set $\Omega_i^1$ and $\Omega_i^2$. As a result, by selecting appropriate parameters, information from uncertain samples can be effectively exploited. Because feature noise around the decision boundary affects classification performance, selecting appropriate parameters $\tau_1, \tau_2, \epsilon_1$, and $\epsilon_2$ can reduce the effect of feature noise. Now, we transform the optimization problem (4.1.1) to the dual form to investigate its properties. In this state, we obtain the problem (4.1.1) by introducing the slack variables as follow,

$$\begin{aligned}
\min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}(\|\mathbf{w}\|^2 + b^2) + C\sum_{i=1}^{m}\xi_i \\
\text{s.t.} \quad & \int_{\Omega_i^1}\Big(\tau_1\big(1 - y_i(\mathbf{w}^\top\mathbf{x} + b)\big) - \epsilon_1\Big) f_{X_i}(\mathbf{x})d\mathbf{x} \le \xi_i, \\
& \int_{\Omega_i^2}\Big(-\tau_2\big(1 - y_i(\mathbf{w}^\top\mathbf{x} + b)\big) - \epsilon_2\Big) f_{X_i}(\mathbf{x})d\mathbf{x} \le \xi_i, \\
& \xi_i \ge 0, \qquad i = 1, 2, 3, \ldots, m.
\end{aligned} \tag{4.1.17}$$

In considering the problem (4.1.17), we introduce the Lagrangian function proposed by:

$$\begin{aligned}
L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) = {} & \frac{1}{2}(\|\mathbf{w}\|^2 + b^2) + C\sum_{i=1}^{m}\xi_i \\
& - \sum_{i=1}^{m}\alpha_i\Big(\xi_i - \int_{\Omega_i^1}\Big(\tau_1\big(1 - y_i(\mathbf{w}^\top\mathbf{x} + b)\big) - \epsilon_1\Big) f_{X_i}(\mathbf{x})d\mathbf{x}\Big) \\
& - \sum_{i=1}^{m}\beta_i\Big(\xi_i - \int_{\Omega_i^2}\Big(-\tau_2\big(1 - y_i(\mathbf{w}^\top\mathbf{x} + b)\big) - \epsilon_2\Big) f_{X_i}(\mathbf{x})d\mathbf{x}\Big) \\
& - \sum_{i=1}^{m}\gamma_i\xi_i,
\end{aligned} \tag{4.1.18}$$

where $\boldsymbol{\alpha}, \boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are nonnegative Lagrange multipliers. For simplicity, we define $K_i^1 = \int_{\Omega_i^1}\Big(\tau_1\big(1 - y_i(\mathbf{w}^\top\mathbf{x} + b)\big) - \epsilon_1\Big) f_{X_i}(\mathbf{x})d\mathbf{x}$ and $K_i^2 = \int_{\Omega_i^2}\Big(-\tau_2\big(1 - y_i(\mathbf{w}^\top\mathbf{x} + b)\big) - \epsilon_2\Big) f_{X_i}(\mathbf{x})d\mathbf{x}$. Note that $K_i^1$ and $K_i^2$ can be represented by using the complementary

error function. By consider (4.1.18) and using the Karush–Kuhn–Tucker(KKT) optimal conditions, we get the following equations:

$$\frac{\partial \mathcal{L}}{\partial b} = b + \sum_{i=1}^{m} \alpha_i \frac{\partial K_i^1}{\partial b} + \sum_{i=1}^{m} \beta_i \frac{\partial K_i^2}{\partial b} = 0, \tag{4.1.19}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\xi}} = C - \alpha_i - \beta_i - \gamma_i = 0, \tag{4.1.20}$$

$$\beta_i (K_i^2 - \xi_i) = 0, \tag{4.1.21}$$

$$\gamma_i \xi_i = 0. \tag{4.1.22}$$

The optimal conditions of (4.1.1) are not completely given here. This is due to the fact that we only use these conditions to discuss the properties of the model. Let us define the set $E_{mis} = i : \xi_i \neq K_i^2$, which represents the number of elements in the set $E_{mis}$. Additionally, let $m_1$ denote the number of misclassified uncertain samples. To demonstrate the conditions that need to be satisfied for $m_1$, we present the following proposition.

**Proposition 4.1.2.** *Assume that there exists the solution to the optimization problem of (4.1.17). Then $m_1$ satisfies the following inequality*

$$m_1 \leq \sum_{i=1}^{m} \frac{|1 - erf(f_i^1)|}{2} + m - \frac{|b|}{C}. \tag{4.1.23}$$

*Proof.* For uncertain samples in the set $E_{mis}$, we have $K_i^2 \neq 0$ and $\xi_i \neq 0$. From (4.1.21) and (4.1.22), we obtain $\beta_i \neq 0$ and $\gamma_i \neq 0$. From (4.1.19), we get $\alpha_i = C$. From (4.1.19), we have $b + \sum_{i=1}^{m} \alpha_i \nabla_b K_i^1 + \sum_{i=1}^{m} \beta_i \nabla_b K_i^2 = 0$. Using (4.1.19), we obtain

$$\begin{aligned} |b| &= \left| \sum_{i=1}^{m} \frac{\alpha_i \tau_1 (1 - \operatorname{erf}(f_i^1))}{2} y_i + \sum_{i=1}^{m} \frac{\beta_i \tau_2 (1 - \operatorname{erf}(f_i^2))}{2} y_i \right| \\ &\leq C \sum_{i=1}^{m} \frac{|(1 - \operatorname{erf}(f_i^1))|}{2} + C \sum_{i \notin E_{mis}} \frac{|\,(1 - \operatorname{erf}(f_i^2))|}{2}. \end{aligned}$$

From (4.1.18), $\beta_i$ take in $[0, C]$. The function $erf(x)$ takes value in $[-1, 1]$. Then,

$$|b| \leq C \sum_{i=1}^{m} \frac{|(1 - \operatorname{erf}(f_i^1))|}{2} + mC - m_1 C.$$

Thus, we have

$$m_1 C = C \sum_{i=1}^{m} \frac{|(1 - \operatorname{erf}(f_i^1))|}{2} + mC - |b|,$$

$$m_1 = \sum_{i=1}^{m} \frac{|(1 - \text{erf}(f_i^1))|}{2} + m - \frac{|b|}{C}.$$

□

The proposition 4.1.2 provides the upper bound of misclassified uncertain samples. If $\mathbf{w}$ and $b$ are given, we can see that the upper bound does not explicitly depend on the parameters $\tau_1$ and $\tau_2$. As mentioned previously in this section, the parameters $\tau_1$ and $\tau_2$ will impact sample classification error.

### 4.1.2 Scatter minimization

Scatter minimization can also be used to understand the UGPinSVMs. For uncertain samples, we define the set $S_1 = \{i \quad |\mathbf{w}^\top \mathbf{x}_i + b = 1 + \frac{\epsilon_2}{\tau_2}, y_i = 1\}$. In the projected space defined by $\mathbf{w}$, the scatter of the uncertain samples in the set $S_1$ is quantified by the expression $\sum_{j \in S_1} \int |\mathbf{w}^\top (\mathbf{x} - \mathbf{x}j)| f_{X_j}(\mathbf{x}) d\mathbf{x}$. We get the following equation by using $\mathbf{w}^\top \mathbf{x}_j + b = 1 + \frac{\epsilon_2}{\tau_2}$ and $y_j = 1$:

$$\sum_{j \in S_1} \int |\mathbf{w}^\top (\mathbf{x} - \mathbf{x}_j)| f_{X_j}(\mathbf{x}) d\mathbf{x} = \sum_{j \in S_1} \int |1 - y_j(\mathbf{w}^\top \mathbf{x} + b) + \frac{\epsilon_2}{\tau_2}| f_{X_j}(\mathbf{x}) d\mathbf{x}. \tag{4.1.24}$$

Similarly, we define the set $S_2 = \{i \quad |\mathbf{w}^\top \mathbf{x}_i + b = -1 - \frac{\epsilon_2}{\tau_2}, y_i = -1\}$. The scatter of the uncertain samples in the set $S_2$ is defined by $\sum_{j \in S_2} \int |\mathbf{w}^\top (\mathbf{x} - \mathbf{x}_j)| f_{X_j}(\mathbf{x}) d\mathbf{x}$. We also obtain this expression

$$\sum_{j \in S_2} \int |\mathbf{w}^\top (\mathbf{x} - \mathbf{x}_j)| f_{X_j}(\mathbf{x}) d\mathbf{x} = \sum_{j \in S_2} \int |1 - y_j(\mathbf{w}^\top \mathbf{x} + b) + \frac{\epsilon_2}{\tau_2}| f_{X_j}(\mathbf{x}) d\mathbf{x}. \tag{4.1.25}$$

According to Eqs. (4.1.24) and (4.1.25), we define the following optimization problem

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{2}b^2 + C_1 \sum_{i=1}^{m} \int_{\mathbb{R}^n} |1 - y_i(\mathbf{w}^\top \mathbf{x} + b) + \frac{\epsilon_2}{\tau_2}| f_{X_i}(\mathbf{x}) d\mathbf{x}, \tag{4.1.26}$$

where $C_1$ is a positive constant. A first term of (4.1.26) being expressed within UGPinSVMs (4.1.5) in its mathematically equivalent form, while the absolute value

employed in (4.1.26) is extended to $\mathcal{L}_{\tau_1,\tau_2}^{\epsilon_1,\epsilon_2}$. It is concretely introducing the misclassification term.

$$C_2 \sum_{i=1}^{n} \int_{\mathbb{R}^n} \mathcal{L}_{hinge}\left(1 - y_i(\mathbf{w}^\top \mathbf{x} + b) + \frac{\epsilon_2}{\tau_2}\right) f_{X_i}(\mathbf{x}) d\mathbf{x}$$
$$= C_2 \sum_{i=1}^{n} \int_{\mathbb{R}^n} \max\left\{0, 1 - y_i(\mathbf{w}^\top \mathbf{x} + b) + \frac{\epsilon_2}{\tau_2}\right\} f_{X_i}(\mathbf{x}) d\mathbf{x}, \tag{4.1.27}$$

and

$$C_3 \sum_{i=1}^{n} \int_{\mathbb{R}^n} \mathcal{L}_{hinge}\left(1 - y_i(\mathbf{w}^\top \mathbf{x} + b) - \frac{\epsilon_1}{\tau_1}\right) f_{X_i}(\mathbf{x}) d\mathbf{x}$$
$$= C_3 \max\left\{0, 1 - y_i(\mathbf{w}^\top \mathbf{x} + b) - \frac{\epsilon_1}{\tau_1}\right\} f_{X_i}(\mathbf{x}) d\mathbf{x}, \tag{4.1.28}$$

into (4.1.26), where $C_2$ and $C_3$ are positive parameters. That is

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{2}b^2 + C_1 \sum_{i=1}^{m} \int_{\mathbb{R}^n} |1 - y_i(\mathbf{w}^\top \mathbf{x} + b) + \frac{\epsilon_2}{\tau_2}| f_{X_i}(\mathbf{x}) d\mathbf{x}$$
$$+ C_2 \sum_{i=1}^{n} \int_{\mathbb{R}^n} \mathcal{L}_{hinge}\left(1 - y_i(\mathbf{w}^\top \mathbf{x} + b) + \frac{\epsilon_2}{\tau_2}\right) f_{X_i}(\mathbf{x}) d\mathbf{x}$$
$$+ C_3 \sum_{i=1}^{n} \int_{\mathbb{R}^n} \mathcal{L}_{hinge}\left(1 - y_i(\mathbf{w}^\top \mathbf{x} + b) - \frac{\epsilon_1}{\tau_1}\right) f_{X_i}(\mathbf{x}) d\mathbf{x}. \tag{4.1.29}$$

Our goal is to identify the representation of (4.1.5). The optimization problem of (4.1.5) is obtained if these conditions ($C_1 = C\tau_2$, $C_3 = C\tau_1$ and $C_1 + C_2 = 0$) are achieved. The proposed model, in a way, takes into account both the scatter and classification error of some uncertain samples. When constructing classifiers, we need to ensure that they have minimal classification error and scatter. Eq. (4.1.27) emphasizes the scatter while concentrating on the classification mistake and absolute loss in (4.1.26). In order to achieve a balance between the small scatter and the small classification error, the proposed model makes this approach.

### 4.1.3 Numerical experiments

For this evaluation, we validate the performance of UGPinSVMs for both binary and multi-class datasets. Furthermore, the crop recommendation dataset, which is the online multi-class dataset from the platform Kaggle, is also investigated. Tables 23 and 24 summarize the detailed information of the binary and multi-class datasets, respectively. multi-class datasets, respectively.

**Table 23 Description of binary datasets.**

| Datasets | No. of Sample | No. of Feature |
|---|---|---|
| Heart | 270 | 13 |
| Spectfheart | 267 | 44 |
| Australian | 690 | 14 |
| Ionosphere | 351 | 33 |
| Bupa | 345 | 6 |
| Fertility | 100 | 10 |
| Pima | 768 | 8 |
| Banknote | 1372 | 4 |
| Titanic | 2201 | 41 |
| Breast | 286 | 9 |
| Phoneme | 5404 | 5 |
| Haberman | 306 | 3 |
| Monk-2 | 432 | 7 |
| Monk-3 | 432 | 7 |
| Sonar | 208 | 60 |
| Diabetes | 769 | 9 |

All samples are selected such that the feature is located in $[0, 1]$ before training. The experiments are conducted on Python 3.9.5. on Intel(R) Core(TM) i5-8500 CPU @3.00 GHz with 16 GB of RAM.

**Table 24 Description of multiclass UCI datasets.**

| Datasets | No. of Sample | No. of Feature | No. of Class |
|---|---|---|---|
| Car | 1728 | 6 | 4 |
| Iris | 150 | 4 | 3 |
| Teaching eval. | 151 | 5 | 3 |
| Balance | 625 | 4 | 3 |
| Ecoli | 336 | 8 | 8 |
| Seeds | 210 | 7 | 3 |

To show the noise insensitivity, we let the features of the dataset corrupted by zero mean Gaussian noise. For each feature, The values of 0 (noise-free), 0.05, and 0.1 are selected for the ratio of noise variance to feature variance, represented by $r$. Note that the same noise is given to the training and testing data. The classification performance is analyzed and compared with the ones of SVMsG [22] and UPinSVMs [23]. In the experiments, we explore both linear cases and nonlinear cases using the Gaussian RBF kernel, $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\big(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\big)$, where $\sigma$ is a parameter. Classification accuracy is determined by the standard 10-fold cross-validation.

### 4.1.4 Parameter selection

For all involved models, we use 50 per cent of each dataset and adopt 5-fold cross-validation to select the best parameter. After that, the datasets are sent back to the training datasets. Both binary and multiclass classification use the kernel parameter $\sigma$ tuning from the set $\{10^i, i = -3, -2, -1, 0, 1, 2, 3\}$. The other parameters are set in the following. For binary classification, we set $c \in \{2^i, i = -3, -2, -1, 0, 1\}$ and $\tau_1, \tau_2, \epsilon_1, \epsilon_2 \in \{0.2, 0.4, 0.8, 1\}$ in the linear case. For Nonlinear case, we tuning the parameters $c$ from $\{10^i, i = -4, -3, -2, -1, 0, 1, 2\}$ while $\tau_1, \tau_2, \epsilon_1, \epsilon_2$ are chosen from the range $\{0.2, 0.3, 0.4, 0.5, 0.7, 0.8, 1\}$. For multiclass classification, the parameters are the same as binary classification except $c$. The setting of $c$ is $\{10^i, i = -2, -1, 0, 1, 2\}$.

### 4.1.5 Binary classification

The comparison results for linear and nonlinear cases are shown in Tables 25 and 26, respectively. From Table 25, UGPinSVMs has the most wins over all other models, i.e., 31 out of 45 (68.89%). SVMsG and UPinSVMs outperform UGPinSVMs only on the Haberman datasets. Despite this, our model still achieves results close to being among the top performers. In Monk-3 datasets, our UGPinSVMs win compared to SVMs and UPSVMs. However, it should be noted that the overall accuracy of the Monk-3 is relatively low compared to other datasets, ranging from approximately 43% to 50%. For the nonlinear case in Table 26, UGPinSVMs have the highest rate of wins among all models, i.e., 18 out of 27 (75%). On Bupa datasets, UPinSVMs perform better than UGPinSVMs, but UGPinSVMs perform better than SVMsG. Additionally, although the highest accuracy of Fertility datasets belongs to SVMsG, almost the results of our model are still outperformed.

Figure 25 summarized the average rank of each model for linear and nonlinear cases on binary classification. In the linear case, we obtain $\mathcal{X}_F^2 = 31.9005, F_F = 24.1589$ and the critical value of $F(2, 88)$ for a significance level of 0.05 is 3.1006, so we reject the null hypothesis and conclude that there is a difference the among three classifiers for accuracy. In additional, we consider nonlinear case in Figure 25 and obtain $\mathcal{X}_F^2 = 19.5858$, $F_F = 14.7971$. Therefore, we also reject the null hypothesis, which implies that there is a difference among the three classifiers. As can be seen in Figure 25, UGPinSVMs obtain the smallest rank values in both cases. This implies that our method is clearly superior to SVMsG and UPinSVMs.

**Table 25 The mean (%) and standard deviation of tests accuracy with various noise on the binary dataset are calculated using a linear kernel.**

| Datasets | $r$ | SVMsG | UPinSVMs | UGPinSVMs |
|---|---|---|---|---|
| Australian | 0 | 86.52±2.90 | 86.81±4.17 | **88.12**± 4.19 |
| | 0.05 | 86.67±4.19 | **87.68**±4.16 | 87.25±3.29 |
| | 0.1 | 85.65±4.69 | 86.67±4.39 | **86.81**±3.69 |
| Bupa | 0 | 60.23±8.16 | **61.13**±8.24 | 60.83±8.79 |
| | 0.05 | 59.34±7.34 | 60.83±7.30 | **61.70**±7.94 |
| | 0.1 | **61.39**±6.24 | 59.67±8.22 | 60.25±7.09 |
| Ionosphere | 0 | 83.22±6.30 | 84.93±7.89 | **88.91**±6.86 |
| | 0.05 | 83.21±5.67 | 86.36±7.43 | **88.63**±6.57 |
| | 0.1 | 83.21±5.42 | 85.79±8.18 | **88.63**±5.92 |
| Pima | 0 | 74.20±6.00 | **76.93**±6.27 | 76.28±5.81 |
| | 0.05 | 73.68±6.32 | 75.77±4.93 | **76.28**±5.63 |
| | 0.1 | 73.16±5.74 | 76.02±6.76 | **76.28**±5.52 |
| Phoneme | 0 | 74.39±1.51 | 76.81±1.60 | **77.05**±1.84 |
| | 0.05 | 70.65 ± 2.19 | **77.02** ± 1.57 | 76.87±1.94 |
| | 0.1 | 70.65 ± 2.19 | **76.98**±1.49 | 76.79 ± 1.53 |
| Breast | 0 | 69.24 ± 15.86 | 65.38±15.69 | **71.12**±10.21 |
| | 0.05 | 65.08 ± 14.21 | 67.05 ± 16.59 | **70.76** ± 8.37 |
| | 0.1 | 66.67 ± 13.05 | 66.36 ± 19.59 | **71.36** ± 10.58 |
| Specfheart | 0 | 78.25 ±7.11 | 79.36 ± 7.43 | **79.36** ± 7.61 |
| | 0.05 | 78.99 ± 7.40 | 79.36 ± 6.00 | **79.73** ± 7.43 |
| | 0.1 | 79.36±7.61 | 79.37 ± 5.17 | **79.73** ± 7.97 |
| Heart | 0 | 83.70 ±6.46 | 81.85±8.19 | **84.07**±7.95 |
| | 0.05 | 83.33 ± 4.46 | 78.15±9.29 | **84.07**±7.60 |
| | 0.1 | 82.59±5.51 | 76.67±14.25 | **84.44**±7.73 |
| Titanic | 0 | 77.60±3.31 | 77.96±3.40 | **78.10** ±3.38 |
| | 0.05 | 77.60 ±3.31 | 77.92±3.51 | **78.15**± 3.48 |
| | 0.1 | 77.60±3.31 | 77.65 ± 3.74 | **78.01** ± 3.55 |
| Monk-2 | 0 | 80.10±8.30 | 79.39±6.88 | **80.31** ±8.29 |
| | 0.05 | **80.33** ±8.10 | 79.39±6.37 | 80.31± 8.23 |
| | 0.1 | 79.62±8.03 | 80.31 ± 6.39 | **80.31** ± 6.80 |

**Table 25 (Cont.)**

| Datasets | $r$ | SVMsG | UPinSVMs | UGPinSVMs |
|---|---|---|---|---|
| Sonar | 0 | 74.02±11.66 | **78.43**±11.43 | 77.86 ±9.42 |
| | 0.05 | 73.52 ±10.11 | **77.81**±8.55 | 77.38± 8.32 |
| | 0.1 | 74.05±9.87 | 75.36 ± 9.90 | **78.29** ± 7.77 |
| Monk-3 | 0 | 44.68±5.60 | 45.62±4.36 | **48.86** ±4.85 |
| | 0.05 | 43.09 ± 8.30 | 47.24±5.36 | **49.76**± 7.95 |
| | 0.1 | 45.15±7.62 | 46.78 ± 5.92 | **49.80** ± 6.78 |
| Haberman | 0 | 74.20±5.19 | **74.20**±5.78 | 73.56±5.37 |
| | 0.05 | **73.87**±5.15 | 72.23±5.26 | 73.23±5.05 |
| | 0.1 | **73.88**±5.10 | 73.55±5.95 | 73.56±5.37 |
| Diabetes | 0 | 76.45±5.79 | 75.64±5.52 | **76.55**±5.43 |
| | 0.05 | 75.15±5.43 | 76.15±5.29 | **76.16**±5.97 |
| | 0.1 | 76.45±6.43 | 76.15±5.60 | **76.67**±5.86 |
| Fertility | 0 | **88.00**±7.48 | 87.00±6.40 | **88.00**±7.48 |
| | 0.05 | **88.00**±7.48 | **88.00**±7.48 | **88.00**±7.48 |
| | 0.1 | **88.00**±7.48 | 87.00±6.40 | **88.00**±7.48 |

### 4.1.6 Multi-class datasets

For the $k$-class multi-category classification problem, we utilized the One-to-Rest validation methodology to provide our UGPinSVMs on the multi-class datasets. With the One-to-Rest technique, a hyperplane is required to divide each class from the others simultaneously. This means that the separation considers all points and divides them into two groups: one for class points and one for all others. The classifier in the One-to-Rest approach can use $k$ SVMs.

The graphical of Figure 26 displays the average rank of accuracy for both cases. The critical value of $F(2, 34)$ for a significance level of 0.05 is 3.276. We obtain $\mathcal{X}_F^2 = 12.8844, F_F = 9.47$ and $9.47 > 3.28$, so we reject the null hypothesis and conclude there is a difference among the three classifiers for accuracy on the linear case. In additional, we obtain $\mathcal{X}_F^2 = 8.75$ and $F_F = 5.46$ for nonlinear case. Therefore, we reject the null hypothesis and conclude there is a difference among the

five classifiers. In addition, as shown in Figure 26, the proposed UGPinSVMs had a lower average rank. That is, the proposed UGPinSVMs significantly outperformed another classifier in classification performance in both linear and nonlinear cases on multi-class datasets.

The comparison results for linear and nonlinear cases are shown in Tables 27 and 28, respectively. As can be observed from Table 27, UGPinSVMs exhibit the highest win rate among all methods, 10 out of 18 (55.56%). Regarding the Iris datasets, all three approaches demonstrate identical accuracy with 66.67. In the Balanced dataset, SVMsG achieves the highest accuracy when $r = 0.05$, and our model proves valid when $r = 0.1$, with an accuracy of 91.99. Similarly, in the Seed dataset, SVMsG matches our method with accuracies of 90.00 and 90.48 for $r = 0.05$ and 0.1, respectively. Table 28 with RBF kernel shows that our UGPinSVMs get the highest accuracy in 12 out of 18 cases (66.67%). The performance remained consistent in the car dataset, where it achieved an accuracy of 70.02. However, there is one situation in the Ecoli dataset where UPinSVMs outperformed both our method and SVMsG, attaining an accuracy value of 85.98.

Figure 27 depicts the convergence paths of SVMsG, UPinSVMs, and UG-PinSVMs in terms of the number of iterations. The graphic shows that the proposed model outperformed the other models. The main concept depicted in this figure is that, during the same iteration, our objective function is lower compared to other methods. This demonstrates that UGPinSVMs converge to a good approximation.

**Table 26 The mean (%) and standard deviation of tests accuracy with various noise on the binary dataset are calculated using RBF kernel.**

| Datasets | $r$ | SVMsG | UPinSVMs | UGPinSVMs |
|---|---|---|---|---|
| Bupa | 0 | 70.15±4.26 | **71.93**±7.97 | 70.18±4.68 |
| | 0.05 | 69.86±7.37 | **71.04**±6.89 | 70.47±5.30 |
| | 0.1 | 70.13±6.71 | 70.19±7.26 | **70.47**±5.11 |
| Ionosphere | 0 | 86.63±9.53 | 88.63±6.81 | **89.49**±7.46 |
| | 0.05 | 87.19±7.98 | 87.78±7.89 | **89.50**±7.95 |
| | 0.1 | 86.92±9.08 | 88.91±5.84 | **89.48**±6.18 |
| Pima | 0 | 73.30±7.77 | 73.95±6.61 | **80.37**±9.67 |
| | 0.05 | 73.43±8.34 | 74.21±6.95 | **81.48**±8.76 |
| | 0.1 | 73.82±7.50 | 73.69±6.38 | **78.15**±9.29 |
| Specfheart | 0 | 79.36 ±7.61 | 79.73 ± 12.57 | **80.10** ± 8.22 |
| | 0.05 | 79.36 ± 7.61 | **80.46** ± 10.30 | 79.37 ± 7.65 |
| | 0.1 | **79.37**±11.22 | 77.11 ± 12.76 | 78.68 ± 6.32 |
| Titanic | 0 | 76.28±3.45 | 77.55±2.43 | **77.69** ±2.42 |
| | 0.05 | 75.65 ±2.54 | **77.37**±2.76 | 77.37± 2.64 |
| | 0.1 | 74.60±2.61 | **77.46** ± 2.71 | 77.37 ± 2.62 |
| Monk-2 | 0 | 94.43±2.14 | 93.06±2.53 | **94.68** ±3.12 |
| | 0.05 | 92.58 ±4.97 | 93.30±3.45 | **94.45**± 2.96 |
| | 0.1 | 91.65±5.23 | 91.67 ± 5.40 | **94.92** ± 3.07 |
| Sonar | 0 | 79.86±10.52 | 82.76±9.11 | **83.74** ±7.95 |
| | 0.05 | 78.45 ±7.31 | 80.79±10.67 | **80.88**± 8.74 |
| | 0.1 | 78.00±9.23 | **81.29** ± 10.56 | 80.90 ± 9.23 |
| Banknote | 0 | 94.97±1.71 | 98.25±1.23 | **99.27** ±0.46 |
| | 0.05 | 94.53 ± 1.60 | 97.81±1.66 | **99.20**± 1.55 |
| | 0.1 | 94.39±1.78 | 97.81 ± 1.56 | **99.49** ± 0.57 |
| Fertility | 0 | **88.00**±11.66 | 86.00±11.14 | 84.00 ±9.17 |
| | 0.05 | **88.00** ± 11.66 | 82.00±12.44 | 85.00± 12.04 |
| | 0.1 | **88.00**±11.66 | 87.00 ± 11.00 | 82.00 ± 10.77 |

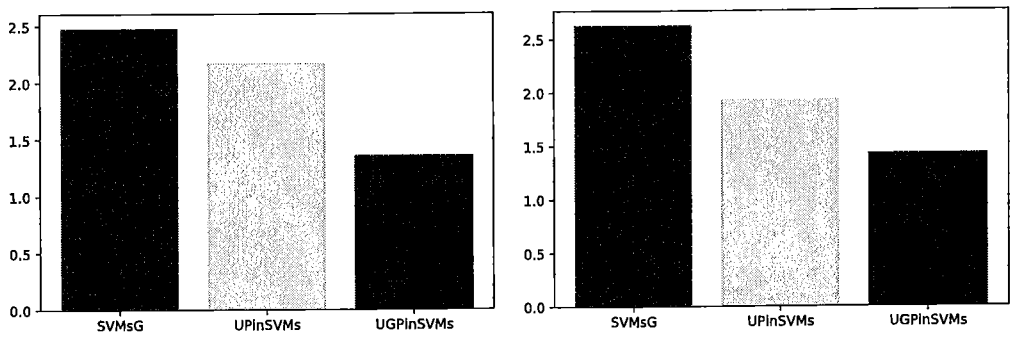



**Figure 25** Average ranks of accuracy on binary-class datasets. The linear and nonlinear cases are arranged from left to right.

**Table 27** Performance evaluation of the proposed model and other models on multi-class datasets with linear kernel.

| Datasets | $r$ | SVMsG | UPinSVMs | UGPinSVMs |
|---|---|---|---|---|
| Car | 0 | 78.76±2.71 | 77.75±3.81 | **78.88**±3.10 |
| | 0.05 | **78.59**±2.68 | 77.49±3.18 | 78.37±1.44 |
| | 0.1 | **79.23**±2.70 | 77.43±3.28 | 77.95±2.22 |
| Iris | 0 | **66.67**±10.75 | **66.67**±10.33 | **66.67**±10.33 |
| | 0.05 | **66.67**±10.75 | **66.67**±10.33 | **66.67**±10.33 |
| | 0.1 | **66.67**±10.75 | **66.67**±10.33 | **66.67**±10.33 |
| Teaching eval. | 0 | 55.00±10.03 | 56.33±9.24 | **57.00**±9.71 |
| | 0.05 | 55.04±11.82 | 53.71±9.44 | **55.67**±8.17 |
| | 0.1 | 54.33±11.16 | 51.71±9.99 | **56.33**±7.06 |
| Balance | 0 | 92.16±2.90 | 91.85±3.71 | **92.17**±3.51 |
| | 0.05 | **92.16**±2.90 | 91.84±2.81 | 92.05±0.31 |
| | 0.1 | **91.99**±2.86 | 91.52±2.95 | **91.99**±2.86 |
| Ecoli | 0 | 80.92±9.33 | 80.62±10.26 | **85.08**±8.40 |
| | 0.05 | 80.33 ±10.33 | 80.03±9.87 | **85.06**±10.04 |
| | 0.1 | 80.32±10.70 | 80.02±9.14 | **83.87**±9.77 |
| Seeds | 0 | 90.00±6.19 | 87.14±6.04 | **90.48**±5.22 |
| | 0.05 | **90.00**±6.19 | 87.62±5.71 | **90.00**±5.41 |
| | 0.1 | **90.48**±5.22 | 87.14±4.29 | **90.48**±5.22 |

**Table 28 Performance evaluation of the proposed model and other models on multi-class datasets with RBF kernel.**

| Datasets | $r$ | SVMsG | UPinSVMs | UGPinSVMs |
|---|---|---|---|---|
| Car | 0 | **70.02**±2.65 | **70.02**±2.65 | **70.02**±2.65 |
| | 0.05 | **70.02**±2.65 | **70.02**±2.65 | **70.02**±2.65 |
| | 0.1 | **70.02**±2.65 | **70.02**±2.65 | **70.02**±2.65 |
| Iris | 0 | 92.00±8.33 | 94.67±6.53 | **95.33**±6.70 |
| | 0.05 | 91.33±7.92 | 91.33±12.31 | **95.33**±6.70 |
| | 0.1 | 91.33±6.70 | 90.67±12.00 | **95.33**±6.70 |
| Teaching eval. | 0 | 59.62±9.95 | 54.29±13.16 | **62.29**±8.73 |
| | 0.05 | 55.00±13.10 | 58.29±13.93 | **62.83**±7.68 |
| | 0.1 | 53.63±9.93 | 52.29±11.14 | **59.62**±8.51 |
| Balance | 0 | 94.08±3.11 | 92.49±3.17 | **99.36**±1.07 |
| | 0.05 | 93.60±3.02 | 92.32±3.15 | **99.52**±0.73 |
| | 0.1 | 93.60±3.11 | 92.32±3.07 | **99.04**±0.79 |
| Ecoli | 0 | 85.96±9.51 | **85.98**±7.12 | 85.96±9.31 |
| | 0.05 | 85.08±7.95 | **85.98**±7.12 | 85.65±10.40 |
| | 0.1 | 85.35±8.93 | **85.98**±7.08 | 85.09±8.21 |
| Seeds | 0 | 91.43±4.67 | 92.38±7.13 | **94.76**±3.96 |
| | 0.05 | 90.48±4.97 | 92.38±7.13 | **94.76**±3.96 |
| | 0.1 | 90.48±4.97 | 92.38±7.13 | **94.76**±3.96 |

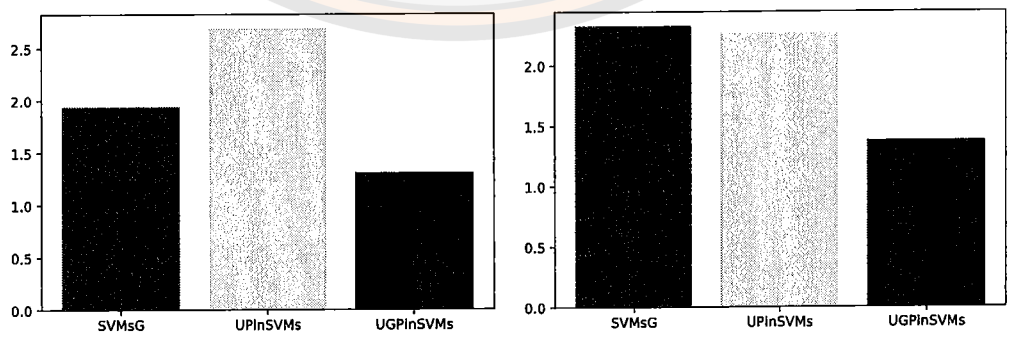
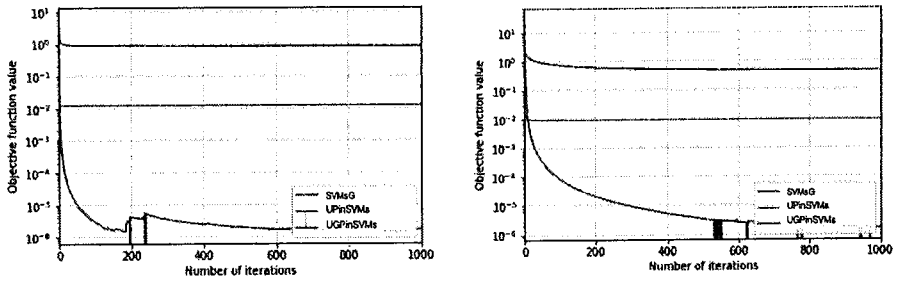
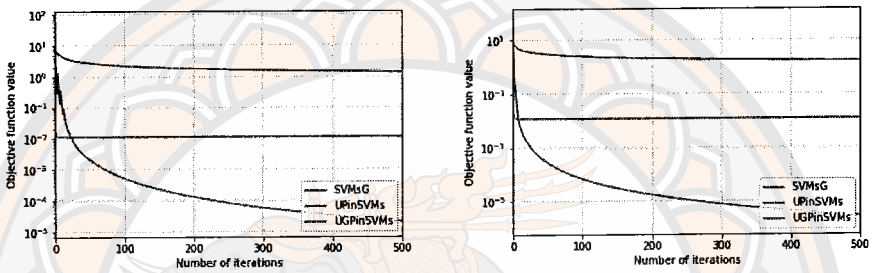



**Figure 26 Average ranks of accuracy on multi-class datasets. The linear and nonlinear cases are arranged from left to right.**




(a) Ionosphere dataset (b) Bupa dataset




(c) Spectfheart dataset (d) Heart dataset

**Figure 27 Convergence curves of trained models for 1,000 iterations for linear cases and 500 iterations for nonlinear cases, (a) and (b) are linear cases, (c) and (d) are nonlinear cases.**

Tables 29 and 30 show the number of wins, draw, and loss for each case. A win is defined as having the highest accuracy compared to the other methods, a loss as not having the highest accuracy, and a draw occurs when two methods exactly attain the highest accuracy. Figure 28 summarizes the win-draw-loss of the sum from Tables 29 and 30 as a percentage. SVMsG-0 and UPinSVMs-0 represent the model SVMsG and UPinSVMs with zero noise data, respectively. It can be seen by the percentage of wins, draws, and losses that UGPin-SVMs outperform the existing method. Our method has a minimum accuracy of 60% and wins every method.

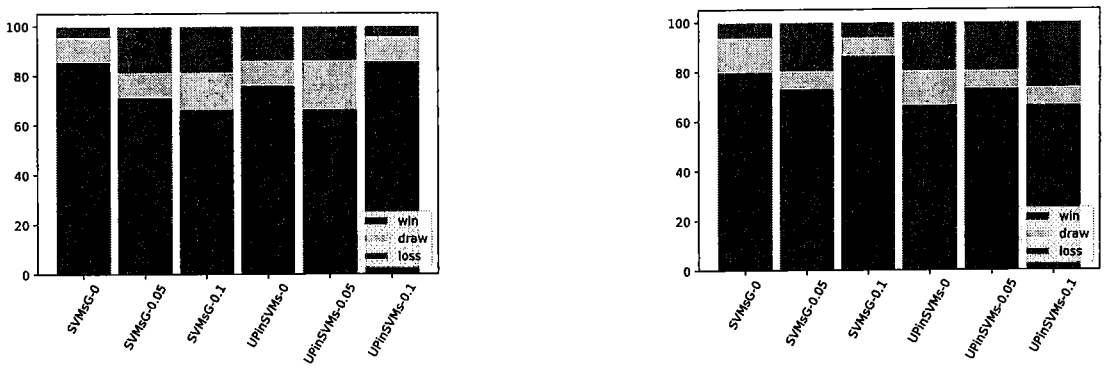



**Figure 28 Win-draw-loss of the sum from Tables 29 and 30. The linear and nonlinear cases are arranged from left to right.**

**Table 29 Win-draw-loss accuracy comparison on binary and multi-class datasets for linear kernel.**

| Datasets | $r$ | SVMsG | UPinSVMs |
|---|---|---|---|
| Binary | 0 | 13-1-1 | 10-1-4 |
| | 0.05 | 12-1-2 | 11-1-3 |
| | 0.1 | 12-1-2 | 13-1-1 |
| Multiclass | 0 | 5-1-0 | 5-1-0 |
| | 0.05 | 2-2-2 | 5-1-0 |
| | 0.1 | 2-3-1 | 5-1-0 |
| Sum | 0 | 18-2-1 | 15-2-4 |
| | 0.05 | 14-3-4 | 16-2-3 |
| | 0.1 | 14-4-3 | 18-2-1 |

**Table 30 Win-draw-loss accuracy comparison on binary and multi-class datasets for RBF kernel.**

| Datasets | $r$ | SVMsG | UPinSVMs |
|---|---|---|---|
| Binary | 0 | 8-0-1 | 7-0-2 |
| | 0.05 | 8-0-1 | 6-1-2 |
| | 0.1 | 7-0-2 | 6-0-3 |
| Multiclass | 0 | 4-2-0 | 4-1-1 |
| | 0.05 | 5-1-0 | 4-1-1 |
| | 0.1 | 4-1-1 | 4-1-1 |
| Sum | 0 | 12-2-1 | 11-1-3 |
| | 0.05 | 13-1-1 | 10-2-3 |
| | 0.1 | 11-1-3 | 10-1-4 |

### 4.1.7 Crop prediction analysis

Precision agriculture is currently popular. It assists farmers in making informed decisions about farming strategies. Machine learning is an important decision support tool for crop yield prediction, advising which crops to grow and what to do during the crop's growing season. Machine learning is present throughout the growing and harvesting process. It starts with crop prediction from soil preparation to seed breeding and water feed measurement and ends with robots picking up the harvest and determining ripeness using computer vision. The crop recommendation dataset has 22,000 samples with 22 classes, such that each class has 100 samples. The graphical illustration of the correlation between the feature of this dataset is shown in Figure 29.

We present the experimental results of the proposed model and compare them to the SVMsG and UPinSVMs models in this section. SVMsG achieves 76.54% accuracy in crop prediction analysis, UPinSVMs 73.96%, and the proposed model 76.93% on the training set. For the testing set, SVMsG achieves 81.14%

accuracy in crop prediction analysis, UPinSVMs 70.25%, and the proposed model 83.86%. The results of the model comparison show that the overall performance of the UGPinSVMs models has improved when compared to the SVMsG and UPinSVMs models, as shown in Table 31.

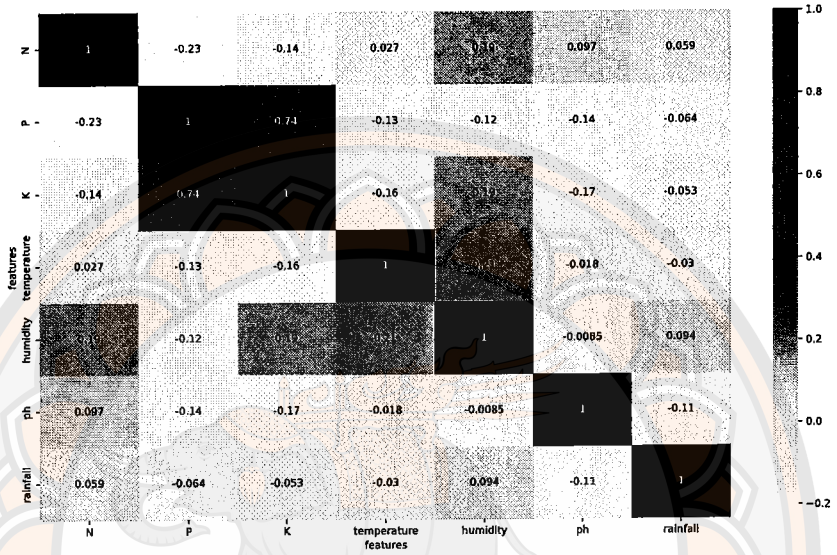



**Figure 29 The correlation between the features of Crop recommendation dataset.**

**Table 31 Classification result on Crop recommendation dataset.**

| | Precision | Recall | F1-score | Accuracy |
|---|---|---|---|---|
| SVMsG | 87.76 | 80.59 | 78.67 | 81.14 |
| UPinSVMs | 67.47 | 71.81 | 64.91 | 70.25 |
| UGPinSVMs | 85.63 | 85.53 | 82.73 | 83.86 |

# CHAPTER VI

# CONCLUSION

In this study, we introduce an innovative method for building a classification model, distinguished by its integration of semi-supervised learning techniques and the handling of data uncertainty through Gaussian distribution analysis. Our experiments cover both linear and non-linear scenarios, with the application of the linear kernel and RBF kernel for analysis in each respective case.

In our initial study, we introduced Lap-PTSVM, a novel method that integrates the pinball loss into the original hinge loss of Lap-TSVM. This represents the first application of the pinball loss for Lap-TSVM, enabling it to address noise sensitivity in data for semi-supervised learning tasks. We provided theoretical analysis to ensure the property of noise insensitivity and scatter minimization, which constitutes a significant advantage of our technique. The performance of Lap-PTSVM is verified through experimental tests and statistical analysis. In our experiments, we explore two scenarios by varying labeled ratios and levels of noise. Regarding the variation in labeled ratios, we find that our proposed outperforms others in 20 out of 30 instances (66.67%). Moreover, in nonlinear cases, 14 out of 21 datasets (66.67%) exhibit the highest prediction accuracy. In terms of varying noise, Lap-PTSVM demonstrates comparable performance to other models in 21 out of 30 cases (70%) in linear scenarios. Similar conclusions hold in nonlinear settings, where Lap-PTSVM achieves better results in 18 out of 36 instances (50%). Finally, our experiments consistently indicate superior performance of Lap-PTSVM, as evidenced by its consistently lowest average rank.

Additionally, we introduced a novel Laplacian twin support vector machine with a generalized pinball loss for semi-supervised classification (LapGPin-TSVM), which is an adaptation of GPin-TSVM enriched with Laplacian Graph-based instance pooling. Our proposed approach enhances model performance by integrating unlabeled data, leading to enhanced generalization, particularly in situations with limited labeled datasets. LapGPin-TSVM establishes a smooth decision

boundary, enhancing robustness in the presence of noise, as guaranteed by theoretical foundations. The solution involves solving a Quadratic Programming Problem (QPP) to determine the classification hyperplanes. Furthermore, this work is considered an extended scenario, encompassing both hinge loss and pinball loss from the previous studies. Experimental results on various UCI benchmarks demonstrate the effectiveness and robustness of LapGPin-TSVM.

Lastly, we introduced a new method termed UGPinSVMs (Uncertain data classification based on Generalized Pinball loss with Support Vector Machines). UGPinSVMs leverage the generalized pinball loss function to enhance performance. The original model we proposed involved the high-dimensional integral problem. In order to make the optimization model become tracable, we transform the original model into the simplified one by using some techniques. We theoretically analyze some properties of the proposed optimization model including noise insensitivity and scatter minimization. To obtain the solution of the proposed model, we adopt stochastic gradient descent (SGD), which is a wide-use and effective algorithm. Experiment results in the various noise indicate that our proposed model is superior to the existing model on both binary and multi-class datasets. The findings indicate that in the context of binary datasets, UGPinSVMs outperform all other models with an accuracy of 31 out of 45 (68.89%). Similarly, in nonlinear scenarios, our proposed demonstrate the highest accuracy among all models, achieving 18 out of 27 wins (75%). When dealing with multiclass datasets, we employed the One-to-Rest validation approach to evaluate the performance of UGPinSVMs. The analysis reveals that in the linear case, UGPinSVMs exhibit the highest accuracy compared to other methods, achieving 10 out of 18 wins (55.56%). In the nonlinear case, our method achieve the highest accuracy in 12 out of 18 instances (66.67%). Furthermore, the statistical analysis demonstrates that UGPinSVMs is significantly better performance than the others.

# REFERENCES

# REFERENCES


1. Vapnik VN. The Nature of Statistical Learning Theory. New York: Springer; 1995.
2. Vapnik VN. An overview of statistical learning theory. IEEE Trans. Neural Netw. 1999;10:988–999.
3. Jayadeva, R. Khemchandani, S. Chandra, Twin support vector machines for pattern classification, IEEE Trans. Pattern Anal. Mach. Intell.2007;29(5):905-910.
4. HUANG, Xiaolin; SHI, Lei; SUYKENS, Johan AK. Support vector machine classifier with pinball loss. IEEE transactions on pattern analysis and machine intelligence.2013;36(5):984-997.
5. XU, Yitian; YANG, Zhiji; PAN, Xianli. A novel twin support-vector machine with pinball loss. IEEE transactions on neural networks and learning systems.2016;28(2):359-370.
6. LI, Kai; LV, Zhen. Smooth twin bounded support vector machine with pinball loss. Applied Intelligence.2021;51:5489-5505.
7. TANVEER, M., et al. Pinball loss twin support vector clustering. ACM Transactions on Multimedia Computing, Communications, and Applications(TOMM).2021;17(2s):1-23.
8. TANVEER, Muhammad; SHARMA, Anurag; SUGANTHAN, Ponnuthurai N. General twin support vector machine with pinball loss function. Information Sciences.2019;494:311-327.
9. EL-ZAHHAR, Mohamed M.; EL-GAYAR, Neamat F. A semi-supervised learning approach for soft labeled data. In: 2010 10th International Conference on Intelligent Systems Design and Applications.IEEE,2010.1136-1141.
10. SINDHWANI, Vikas; KEERTHI, S. Sathiya. Large scale semi-supervised linear SVMs. In: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval.2006.p.477-484.
11. CHAPELLE, Olivier; CHI, Mingmin; ZIEN, Alexander. A continuation method for semi-supervised SVMs. In: Proceedings of the 23rd international conference on Machine learning.2006.p.185-192.

12. BELKIN, Mikhail; NIYOGI, Partha. Laplacian eigenmaps and spectral techniques for embedding and clustering. Advances in neural information processing systems,2001,14.

13. MELACCI, Stefano; BELKIN, Mikhail. Laplacian Support Vector Machines Trained in the Primal. Journal of Machine Learning Research.2011;12(3):)1149-1184.

14. QI, Zhiquan; TIAN, Yingjie; SHI, Yong. Laplacian twin support vector machine for semi-supervised classification. Neural networks.2012;35:46-53.

15. ZHOU, Jia-Bin, et al. Intuitionistic fuzzy Laplacian twin support vector machine for semi-supervised classification. Journal of the Operations Research Society of China.2022;10(1):89-112.

16. BAI, Lan, et al. Safe intuitionistic fuzzy twin support vector machine for semi-supervised learning. Applied Soft Computing. 2022;123:108906.

17. RASTOGI, Reshma; PAL, Aman; CHANDRA, Suresh. Generalized pinball loss SVMs. Neurocomputing.2018;322:151-165.

18. PANUP, Wanida; RATIPAPONGTON, Wachirapong; WANGKEEREE, Rabian. A novel twin support vector machine with generalized pinball loss function for pattern classification. Symmetry.2022;14(2):289.

19. SHARMA, Krishna Kumar; SEAL, Ayan. Outlier-robust multi-view clustering for uncertain data. Knowledge-Based Systems.2021;211:106567.

20. PELISSARI, Renata, et al. Techniques to model uncertain input data of multi-criteria decision-making problems: a literature review. International Transactions in Operational Research.2021;28(2):523-559.

21. XIE, Zongxia; XU, Yong; HU, Qinghua. Uncertain data classification with additive kernel support vector machine. Data & Knowledge Engineering.2018;117:87-97.

22. TZELEPIS, Christos; MEZARIS, Vasileios; PATRAS, Ioannis. Linear maximum margin classifier for learning from uncertain data. IEEE transactions on pattern analysis and machine intelligence.2017;40(12):2948-2962.

23. LIANG, Zhizheng; ZHANG, Lei. Support vector machines with the $\epsilon$-insensitive pinball loss function for uncertain data classification. Neurocomputing.2021;457:117-127.

24. WU, Qing, et al. Least squares twin support vector machine based on manifold-based within-class scatter. In: 2019 IEEE Symposium Series on Computational Intelligence (SSCI).IEEE,2019.p.2897-2902.

25. Friedberg, Stephen H., Arnold J. Insel, and Lawrence E. Spence.Linear algebra. Essex.NJ.USA:Pearson;2014.

26. Dhara, Anulekha, and Joydeep Dutta. Optimality conditions in convex optimization: a finite-dimensional view.United States:CRC Press;2011.

27. Kroese, Dirk P., Zdravko Botev, and Thomas Taimre. Data science and machine learning: mathematical and statistical methods.United States:Chapman and Hall/CRC;2019.

28. TIKHONOV, Andrei Nikolaevich; ARSENIN, V.I.A.K. Solutions of ill-posed problems. .University of Michigan:Winston;1977.

29. WU, Qing, et al. Least squares twin support vector machine based on manifold-based within-class scatter. In: 2019 IEEE Symposium Series on Computational Intelligence (SSCI).IEEE,2019.p.2897-2902.

30. TANVEER, M.; TABISH, M.; JANGIR, Jatin. Sparse pinball twin bounded support vector clustering.IEEE Transactions on Computational Social Systems.2021;9(6):1820-1829.

31. JAIN, Sambhav; ROY, Shuvo Saha; RASTOGI, Reshma. Neo-twin support vector machines for pattern classification. In: 2022 International Conference on Decision Aid Sciences and Applications (DASA).IEEE,2022. p.347-351.

32. TANVEER, Mohammad, et al. Comprehensive review on twin support vector machines. Annals of Operations Research.2022:1-46.

33. Abramowitz, Milton, and Irene A. Stegun, eds. Handbook of mathematical functions with formulas, graphs, and mathematical tables.US Government printing office. Washington, DC :US Government printing office,1948.

34. GARCÍA, Salvador, et al. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. Information sciences.2010;180(10):2044-2064.

35. SHAO, Yuan-Hai, et al. Improvements on twin support vector machines. IEEE transactions on neural networks.2011;22(6):962-968.

36. KHEMCHANDANI, Reshma; JAYADEVA; CHANDRA, Suresh. Optimal kernel selection in twin support vector machines. Optimization Letters.2009;3:77-88.

37. DEMSAR, Janez. Statistical comparisons of classifiers over multiple data sets. The Journal of Machine learning research.2006;7:1-30.

38. Hsu, Chih-Wei et al. A Practical Guide to Support Vector Classication.2008.

39. DAMMINSED, Vipavee; PANUP, Wanida; WANGKEEREE, Rabian. Laplacian Twin Support Vector Machine With Pinball Loss for Semi-Supervised Classification.IEEE Access. 2023;11:31399-31416.