A NOVEL SUPPORT VECTOR MACHINE FOR SOLVING CLASSIFICATION PROBLEM WITH LARGE SCALE DATA SETS

DAWRAWEE MAKMUANG

A Thesis Submitted to the Graduate School of Naresuan University
in Partial Fulfillment of the Requirements
for the Master of Science Degree in Mathematics
June 2020

This thesis entitled "A novel support vector machine for solving classification problem with large scale data sets"

by Dawrawee Makmuang

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

Oral Defense Committee

..... Chair

(Associate Professor Cholwich Nattee, Ph.D.)

..... Advisor

(Associate Professor Rabian Wangkeeree, Ph.D.)

..... Internal Examiner

(Associate Professor Narin Petrot, Ph.D.)

Approved

.....

(Professor Paisarn Muneesawang, Ph.D.)

Dean of the Graduate School

9 JUN 2020

ACKNOWLEDGEMENT

First of all, I would like to thank my advisor, Associate Professor Rabian Wangkeeree, whose motivation, guidance, encouragement and also helped me in doing a lot of research.

I would also like to thank the Science Achievement Scholarship of Thailand for cost support throughout my master degree study.

Furthermore, I am grateful to my family for their support and encouragement throughout my studies and also like to honestly thank my friends for their encouragement and help while I am doing this thesis.

Finally, I would also like to thank the Department of Mathematics, Faculty of Science, Naresuan University for hospitality and materials for my study and research period. Moreover, I would like to honestly thank Graduate School, Naresuan University for supporting my study in my Master's Degree.

Dawrawee Makmuang

Title	A NOVEL SUPPORT VECTOR MACHINE FOR SOLVING CLASSIFICATION PROBLEM WITH LARGE SCALE DATA SETS
Author	Dawrawee Makmuang
Advisor	Associate Professor Rabian Wangkeeree, Ph.D.
Academic Paper	Thesis M.S. in Mathematics Naresuan University, 2019
Keywords	Generalized pinball loss function, Large scale problems, Stochastic gradient descent method, Twin parametric Support Vector Machine.

ABSTRACT

Twin Parametric Support Vector Machine (TPSVM) is a technique designed to solve data classification problems. This is done by generating two nonparallel hyperplanes through a pair of quadratic programming problems (QPPs). The conventional TPSVM technique is based on the hinge loss function. This leads to its sensitivity to noises. In this thesis, we present a stochastic gradient descent method twin parametric support vector machine using the generalized pinball loss function (SG-TPSVM). Our proposed technique aims to improve the noise robustness and the scalability of the TPSVM. Moreover, we also conduct a number of numerical experiments to evaluate the proposed techniques. The experimental results show that our method is noise insensitive and can handle large scale problems.

LIST OF CONTENTS

Chapter Page

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

Table **Page**

CHAPTER I

INTRODUCTION

Support vector machine (SVM) [1] is a popular supervised binary classification algorithm based on statistical learning theory. Within a few years after its introduction the SVM has already outperformed most other systems in a wide variety of applications like financial forecasting [2], human activity recognition [3], bioinformatics [4], and financial regression [5, 6, 7, 8], and so forth.

As an illustration, we now give the following interesting example (diagnosis of heart disease) of binary classification problem: Assume that diastolic (blood) pressure and the level of cholesterol are strong determinants of heart disease. Ten patients' clinical records are listed in Table 1.1. Here, $y_i = 1$ indicates that the i-th patient belongs to positive class having cardiac disease; $y_j = -1$ (not 0) indicates that the j-th patient belongs to negative class having no cardiac disease.

Table 1 Clinical records of 10 patients.

Patient number	Diastolic pressure	Cholesterol level	Having heat disease
1	73	150	$y_1 = -1$
2	85	165	$y_2 = 1$
⋮	⋮	⋮	⋮
10	110	190	$y_{10} = 1$

The problem is, given the diastolic pressure and the level of cholesterol for a new patient, how to deduce whether the patient has heart disease or not. The Support vector machine technique [1] focuses on maximizing the distance between two bounding hyperplanes. It is generally formulated as a convex quadratic programming problem (QPP).

On the one hand, a main challenge for the standard SVM is the high computational complexity of training samples, i.e. $O(m^3)$, where m is a total number of training samples, due to the standard SVM solved a single larger-sized convex quadratic programming problem (QPP) to find an optimal separating hyperplane. In contrast to the aforementioned idea of generate separating hyperplane in SVMs, Jayadeva [10]

proposed a twin support vector machine (TSVM) that generating two nonparallel hyperplanes such that each hyperplane is closer to one of two classes and is at least one far from the other classes. The strategy of TSVM aimed at solving a pair of smaller sized QPPs, instead of solving a large one as in the classical SVM. Therefore, it makes the computational time of TSVM approximately four times faster than the standard SVM in theory. Many various of the TSVMs have been proposed, such as least square twin support vector machine [11, 12], twin support vector machine regression [13, 14], twin parametric support vector machine (TPSVM) [15, 16], etc. Although the various forms of TSVM works faster than the standard SVM, these are not able to handle a very large number of data samples during training as solving the corresponding QPP becomes infeasible. This is due to the computation of the inverse of a large matrix, in Lagrangian dual problems of TSVMs, which needed for all dual solutions. In order to deal with the large scale problem, many improvements were proposed, such as, for SVM, sequential minimal optimization, coordinate decent method, trust region Newton and stochastic gradient descent algorithm (SGD) in [17, 18, 19, 20, 21], and for TSVMs, successive overrelaxation technique, Newton-Armijo algorithm, and dual coordinate decent method in [22, 23, 24].

Stochastic gradient descent and its variants are versatile techniques that have proven invaluable as a learning algorithms for large datasets. The best advice for a successful application of these techniques is to perform small-scale experiments with subsets of the training data, and to pay a ruthless attention to the correctness of the gradient computation. In the spirit of the stochastic gradient algorithm (SGD), Wang [25] recently proposed a stochastic gradient twin support vector machine (SG-TSVM). This technique partitions a large scale problem into a series of subproblems by stochastic sampling with a suitable size. The SG-TSVM has been shown to be the fastest method among the TSVM-type classifiers for large scale problems.

On the other hand, the common loss function used in SVMs and TSVMs model are hinge loss functions, which makes it essentially sensitive to noise, and is not stable for resampling. The SVM model with pinball loss function (Pin-SVM) was proposed by Huang [26] to treat noise sensitivity and instability to re-sampling. However, the Pin-SVM loses sparsity needs to be corrected. To achieve sparsity, they also proposed a modified ϵ-insensitive zone into the Pin-SVM. Although ϵ-insensitive zone Pin-SVM improves loses sparsity of Pin-SVM, its formulation requires the value of ϵ to be specified beforehand and therefore a bad choice may effect its performance.

Taking motivation from these developments, Reshma [27] proposed a modified (ϵ_1, ϵ_2)-insensitive zone Pin-SVM, called generalized pinball loss SVM. This generalized pinball loss SVM model is noise-insensitive as well as sparse in the solution obtained. Nevertheless, compared with the TSVMs, the generalized pinball loss SVM still needs to solve a single large QPP, which leads to a higher computational complexity and not capable to solve large scale problem.

In order to overcome the above-mentioned limitations of large scale problem and inspired by the studies of the TSVMs and the generalized pinball loss function, in this thesis, our aim is to propose a novel twin parametric support vector machine model by using the generalized pinball loss function. Further, we propose to use the stochastic gradient descent algorithm for computing the solution to the twin parametric support vector machine model by using the generalized pinball loss function (SG-TPSVM). The proposed technique is an efficient algorithm for real-world datasets, especially large-scale ones. Moreover, we show that the sequence generated by our SG-TPSVM is almost surely convergent. Finally, we show, by a number of numerical experiments, that the proposed SG-TPSVM approach outperforms the existing approaches in term of accuracy. The results also show the robustness of the proposed approach on noises. The layout of the thesis is as follows.

Chapter II. We have briefly explained about the related works, theorems and definitions

Chapter III. We present the SG-TPSVM with the theoretical analysis.

Chapter IV. We reports experimental results on several machine learning benchmark datasets.

Chapter V. The conclusion of this thesis is presented.

CHAPTER II

PRELIMINARIES

In this chapter is to review related definitions and theorems that will be used in the later chapter, including related methods about a support vector machines for binary classification problems. Throughout this thesis we consider a binary classification in n-dimensional Euclidean space $\mathbb{R}^n$, and $\|\cdot\|$ denote the Euclidean norm of a vector in $\mathbb{R}^n$. All vectors are considered to be column vectors which can be transposed to a row vector by the superscript $\top$. For $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$, the inner product of x, y are given by $\langle x, y\rangle = x_1y_1 + x_2y_2 + \cdots + x_ny_n$ and the norm of x is given by $\|x\| = \sqrt{\langle x, x\rangle}$.

2.1 Definitions, theorems and notations

Definition 2.1.1. A sequence $\{x_k \in \mathbb{R} : k = 1, 2, \ldots\}$ or simply $\{x_k\} \subset \mathbb{R}$ is called **converge** if there exist $x \in \mathbb{R}$ and $k_\varepsilon \in \mathbb{N}$ (that depend on ε) such that for every $\varepsilon > 0$, we have

$$|x_k - x| < \varepsilon, \quad \forall k \geq k_\varepsilon.$$

The scalar x is called the limit of $\{x_k\}$, and the sequence $\{x_l\}$ is said to be converge to x. Symbolically, it is expressed as

$$x_k \to x \quad \text{or} \quad \lim_{l \to \infty^+} x_k = x.$$

Definition 2.1.2. A scalar sequence $\{x_k\}$ is said to be bounded above (resp. below) if there exists some scalar α such that $x_k \leq \alpha$ (resp. $x_k \geq \alpha$) for all $k \in \mathbb{N}$. It is said to be bounded if it is bounded above and bounded below. The sequence $\{x_k\}$ is said to be monotonically nonincreasing (resp. nondecreasing) if $x_k + 1 \leq x_k$ (resp. $x_k + 1 \geq x_k$) forall $k \in \mathbb{N}$. If $x_k \to x$ and $\{x_k\}$ is monotonically nonincreasing (resp. nondecreasing)

Theorem 2.1.3. *A monotone increasing sequence bounded from above is convergent and a monotone decreasing sequence bounded from below is convergent.*

Definition 2.1.4. (Cauchy Criterion for Series) The series $\sum x_k$ is called a convergent if for every $\varepsilon > 0$ there exists $M_\varepsilon \in \mathbb{N}$ such that if $l > k > M_\varepsilon$, then

$$|x_{k+1} + x_{k+2} + \cdots + x_l| < \varepsilon.$$

Definition 2.1.5. The series $\sum x_k$ is said to be **absolutely convergent** if the series $\sum |x_k|$ is convergent.

Theorem 2.1.6. *If a series in $\mathbb{R}$ is absolutely convergent, then it is convergent.*

Proof. Suppost that $\sum |x_k|$ is convergent. By Cauchy Criterion for Series's definition, we have given $\varepsilon > 0$ there exists $M_\varepsilon \in \mathbb{N}$ such that if $l > k \geq M_\varepsilon$, then

$$|x_{k+1}| + |x_{k+2}| + \cdots + |x_l| < \varepsilon.$$

By The triangle Inequality, we have

$$|x_{k+1} + x_{k+2} + \cdots + x_l| < \varepsilon.$$

Since $\varepsilon > 0$ is arbitrary, this implies that $\sum x_k$ is convergent. □

Definition 2.1.7. A sequence $\{\mathbf{x}_k\}$ of vectors in $\mathbb{R}^n$ is said to converge to some $\mathbf{x} \in \mathbb{R}^n$ if the i-th component of $\mathbf{x}_k$ converges to the i-th component of $\mathbf{x}$ for every $i = 1, 2, \ldots, n$. We use the notations $\mathbf{x}_k \to \mathbf{x}$ or $\lim_{k\to+\infty} \mathbf{x}_k = \mathbf{x}$ to indicate convergence for vector sequences as well.

Next we go through the notions of Convergence for Sequences of Random Variables. Consider a sequence of random variables $\{X_k\}$ that is defined on an underlying sample space S. For simplicity, let us assume that S is a finite set, so we can write

$$S = \{s_1, s_2, \ldots, s_d\}.$$

Then, for each X_k is a function from S to the set of real number Thus, we may write

$$X_k(s_j) = x_{kj}, \quad \text{for } j = 1, 2, \ldots, d.$$

Definition 2.1.8. Let $\{X_k\}$ be a sequence of random variables defined on a sample space S. A sequence $\{X_k\}$ is said to almost surely convergent to a random variables X defined on S if the sequence of real numbers $\{X_k(s_j)\}, j = 1, 2, \ldots, d$ converges to $X(s_j), j = 1, 2, \ldots, d$ almost surely, i.e., if there exists a zero-probability event E such that

$$\Big\{ s \in S : \{X_k(s)\} \text{ does not converge to } X(s) \Big\} \subset E.$$

X is called the almost sure limit of $\{X_k\}$. Symbolically, it is expressed as

$$X_k \xrightarrow{a.s.} X.$$

Definition 2.1.9. Let $\{\mathbf{X}_k\}$ be a sequence of random vectors defined on a sample space S, where each random vector $\mathbf{X}_k \in \mathbb{R}^n$. A sequence $\{\mathbf{X}_k\}$ is said to almost surely convergent to a random vector $\mathbf{X}$ defined on S if the i-th component of $\mathbf{X}_k(s_j)$ converges almost surely to the i-th component of the random variable $\mathbf{X}(s_j)$ for every $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, d$. We use the notations $\mathbf{X}_k \xrightarrow{a.s.} \mathbf{X}$ to indicate almost surely convergent for random vector sequences as well.

Example 2.1.10. *(Almost surely convergent) Let the sample space be* $S = [0, 1]$, *i.e. the sample space is the set of all real numbers between* 0 *and* 1. *Sub-intervals of* $[0, 1]$ *are assigned a probability equal to their length:*

$$\Pr([\mathrm{a}, \mathrm{b}]) = \mathrm{b} - \mathrm{a} \quad \forall 0 \leq \mathrm{a} \leq \mathrm{b} \leq 1.$$

Define a sequence of random variables $\{X_k\}$ *as follows:*

$$X_k(s) = s^k \quad \forall s \in S.$$

Define a random variable X *as follows:*

$$X(s) = 0 \quad \forall s \in S.$$

Show that the sequence $\{X_k\}$ *converge almost surely to* X.

Proof. For a fixed sample point $s \in [0, 1)$, we have

$$\lim_{k\to\infty} X_k(s) = \lim_{k\to\infty} s^k = 0.$$

On the one hand, for $s = 1$, we have

$$\lim_{k\to\infty} X_k(s) = \lim_{k\to\infty} s^k = \lim_{k\to\infty} 1^k = 1.$$

This implies that, for $s = 1$, the sequence of random variables $\{X_k\}$ does not converge to X, i.e.,

$$\lim_{k\to\infty} X_k(s) \neq X(s).$$

Therefore,

$$\Pr\left(\left\{s \in S : \{X_k(s)\} \text{ does not converge to } \{X(s)\}\right\}\right) = \Pr\left(\{1\}\right) = 1 - 1 = 0$$

Thus, the sequence $\{X_k(s)\}$ converges almost surely to X. □

Now we move on to a priori knowledge of a real-valued function that involve the differentiability and the convexity of the function. We begin with by recalling the partial derivative.

Definition 2.1.11. Let $f : \mathbb{R}^n \to \mathbb{R}$ be some function, fix some $\mathbf{x} \in \mathbb{R}^n$. If the limit

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

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 $\frac{\partial f}{\partial x_i}(\mathbf{x})$. A function f is called **continuously differentiable** over $\mathbb{R}^n$ if all partial derivatives exist and are continuous on $\mathbb{R}^n$.

Definition 2.1.12. *A* set $S \subset \mathbb{R}^n$ is called a **convex set** if for any $\mathbf{x}_1, \mathbf{x}_2 \in S$and any $\lambda \in [0, 1]$, we have

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

Theorem 2.1.13. *Let A and B be convex sets, then $A \cap B$ is convex.*

Definition 2.1.14. 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 any $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n$ 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 $\lambda \in [0, 1]$,

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

Theorem 2.1.15. *If two functions are convex, the composition of the functions is convex.*

Definition 2.1.16. Let $f : C \to \mathbb{R}$ be a convex function defined over a convex set $C \subseteq \mathbb{R}^n$. The function f is called a **strongly convex function** with parameter $\rho > 0$ if for any $\mathbf{x}_1, \mathbf{x}_2 \in C$ and $\lambda \in [0, 1]$,

$$f(\lambda \mathbf{x}_1 + (1 - \lambda)\mathbf{x}_2) \leq \lambda f(\mathbf{x}_1) + (1 - \lambda) f(\mathbf{x}_2) - \frac{\rho}{2}\lambda(1 - \lambda)\|\mathbf{x}_1 - \mathbf{x}_2\|^2.$$

Definition 2.1.17. 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 $\partial f(\bar{\mathbf{x}})$.

Theorem 2.1.18. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a deffirentiable convex function at $\bar{\boldsymbol{x}}$. Then $\partial f(\bar{\boldsymbol{x}}) = \{\nabla f(\bar{\boldsymbol{x}})\}$, where $\nabla f(\boldsymbol{x}) = \left[\frac{\partial f(\boldsymbol{x})}{\partial x_1}, \frac{\partial f(\boldsymbol{x})}{\partial x_2}, \ldots, \frac{\partial f(\boldsymbol{x})}{\partial x_n}\right]^\top$.*

Theorem 2.1.19. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a deffirentiable. Then f is convex function if and only if*

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

Theorem 2.1.20. *Let $f : C \to \mathbb{R}$ be a twice continuously deffirentiable function defined over a convex set $C \subseteq \mathbb{R}^n$. Then f is convex if and only if $\nabla^2 f(\boldsymbol{x}) \succeq 0$ for all $\boldsymbol{x} \in C$.*

Theorem 2.1.21. *Let $f : C \to \mathbb{R}$ be a convex function defined over a convex set $C \subseteq \mathbb{R}^n$. Let $\boldsymbol{x}^* \in C$ be a local minimum of f over C. Then $\boldsymbol{x}^*$ is a global minimum of f over C.*

Corollary 2.1.22. *Let $f : C \to \mathbb{R}$ be a convex function defined over a convex set $C \subseteq \mathbb{R}^n$. Then, the set of all global minimizer of f over C is a convex set.*

Theorem 2.1.23. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable convex function. Then $\nabla f(\boldsymbol{x}^*) = 0$ if and only if $\boldsymbol{x}^*$ is a global minimum point of f over $\mathbb{R}^n$.*

We now define a general constrained optimization problem and the specific properties associated to convex constrained optimization problems.

Definition 2.1.24. Let $f, g_i : X \to \mathbb{R}$, $i = 1, \ldots, m$ be a convex function defined over a set $X \subseteq \mathbb{R}^n$. Then, a convex constrained optimization problem has the form:

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

Proposition 2.1.25. *The level set $\{x \in$ convex set $C : g(x) \leq 0\}$ for a constrained optimization problem is convex if the constraints are linear.*

Definition 2.1.26. 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:

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

where the variables $\boldsymbol{\alpha}_i$ are known as the Lagrange multipier or dual variables with $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)^\top$.

Definition 2.1.27. The dual (optimization) problem associated to the constrained optimization problem is

$$\begin{aligned} &\max_{\boldsymbol{\alpha}} \quad \inf_{\mathbf{x}\in X} \mathcal{L}(\mathbf{x}, \boldsymbol{\alpha}), \\ &\text{subject to} \quad \boldsymbol{\alpha} \geq 0. \end{aligned} \tag{2.1.2}$$

Definition 2.1.28. Assume that $\text{int}(X) \neq \emptyset$. Then Slater's condition is defined as

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

Next theorem we give necessary and sufficient optimality conditions of convex constrained optimization problems.

Theorem 2.1.29. *Assume that $f, g_i : X \to \mathbb{R}$, $i = 1, \ldots, m$ are convex functions and additionally the constraints are satified the Slater's condition. Then $\boldsymbol{x}^*$ is a solution of the constrained program if and only if there exists $\boldsymbol{\alpha} \geq 0$ such that,*

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

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

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

The above condition are known as the KKT condition.

Let us finally of this section turn to the Big O notation. Definition of the Big O notation is given as follows

Definition 2.1.30. Let $f : X \to \mathbb{R}$ be a function defined over a set $X \subseteq \mathbb{R}^n$. Let g be a real valued function with $g(\mathbf{x}) \neq 0$. Then $f(\mathbf{x}) = O(g(\mathbf{x}))$, if there exist $c > 0$ and $\mathbf{n}_0 > 0$ such that

$$|f(\mathbf{x})| \leq c|g(\mathbf{x})| \quad \forall \mathbf{x} \geq \mathbf{n}_0.$$

The order of computational complexity can be interpreted using the notation of Big O. Computational complexity is a theory of how difficult it is to answer a problem. This difficulty is measured by the number of operations an algorithm needs to find the correct answer to the problem in the worst case.

Example 2.1.31. *Let* $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ *and* $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$ *If we compute the inner product of* $\boldsymbol{x}$,$\boldsymbol{y}$, *we need* n *multiplication and* $n-1$ *additions, resulting in a total of* $2n-1$ *arithmetic operation. That is, the computational complexity of* $\langle x, y \rangle$ *is* n, *since there exist* $c = 2$ *and* $n_0 = 1$ *such that*

$$2n - 1 \leq 2(n) \quad \forall n \geq 1$$

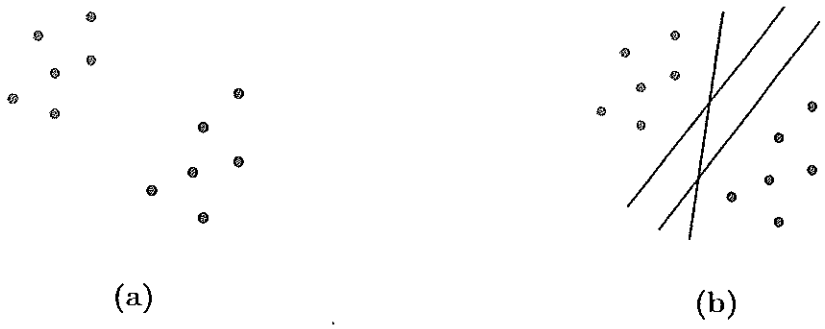
So, $f(n) = 2n - 1 = O(n)$.

2.2 Related works

In this section is to review related methods for binary classification problems. We denote the training dataset is defined as $D = \{(\mathbf{x}_i, y_i) \in \mathbb{R}^n \times \{1, -1\} | i = 1, 2, 3, \ldots, m\}$ and the matrix $X = [x_1, x_2, \ldots, x_m]^\top \in \mathbb{R}^{m \times n}$. Below, we give a brief outline of several related methods.

2.2.1 Support vector machine

Let us now discuss the support vector machine to deal with binary classification problem. First, we discuss support vector machine, in which training data are linearly separable in the input space. Then we present support vector machine to the case where training data are non linearly separable and map the input space into the high-dimensional feature space to enhance linear separability in the feature space. Let us consider, if we assume that the training dataset T can be linearly separated. It mean that its can be separated by a linear classifiers, or hyperplanes correctly such as illustrated by Figure 1. The definition of linearly separable problem is as follows.



(a) (b)

Figure 1: Illustration of linearly separable.

Definition 2.2.1. The training dataset $D = \{(\mathbf{x}_i, y_i) \in \mathbb{R}^n \times \{1, -1\} | i = 1, 2, \ldots, m\}$ is called a linearly separable if there exist $\mathbf{w} \in \mathbb{R}^n, b \in \mathbb{R}$ and $\varepsilon \geq 0$ such that for every sample $i = 1, 2, \ldots, m\}$:

$$\begin{cases} \mathbf{w} \cdot \mathbf{x}_i + b > \varepsilon, & y_i = 1 \\ \mathbf{w} \cdot \mathbf{x}_i + b \leq \varepsilon, & y_i = -1 \end{cases},$$

where $\mathbf{w}$ is called the weight vector and b is referred to as the bias.

However, to separate the two classes of the training data, there are several such separating hyperplanes (see Figure 1b). Note that the distance between the separating hyperplane and the training data sample nearest to the hyperplane is called the margin as shown Figture 2. It is reasonable to select the separating hyperplanes which make the margin maximal and is thus known as the maximum margin hyperplane.

To do this we use the distance from point to the hyperplane formular. In this formular, if we let $H(\mathrm{a}, b) = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}^\top \mathbf{x} + b = 0\}$ where $0 \neq \mathbf{a} \in \mathbb{R}^n, b \in \mathbb{R}$ and let $\mathbf{y} \in \mathbb{R}^n$, then the distance from point $\mathbf{y}$ to hyperplane $H(\mathbf{a}, b)$ is given by

$$d_{H(\mathbf{a},b)}(\mathrm{y}) = \frac{|\mathbf{a}^\top \mathbf{y} + b|}{\|\mathbf{a}\|}$$

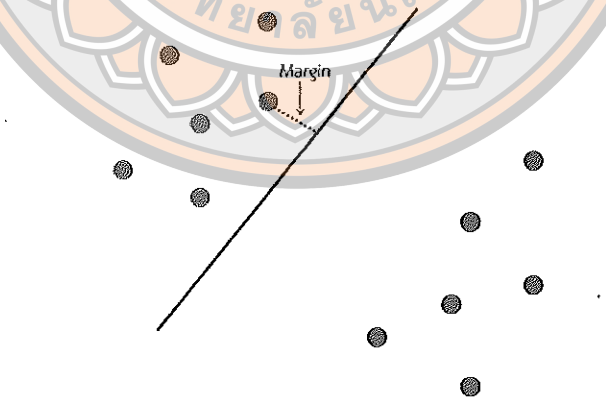


Figure 2: Illustration of margin

Therefore the idea of maximum margin leads to the following optimization problem

$$\max_{\mathbf{w},b} \min_{i=1,2,\ldots,m} \frac{|\mathbf{w}^\top \mathbf{x}_i + b|}{\|\mathbf{w}\|}, \tag{2.2.1}$$

$$\begin{aligned} \text{subject to} \quad & (\mathbf{w}^\top \mathbf{x}_i + b) > 0, \quad i : y_i = 1, \\ & (\mathbf{w}^\top \mathbf{x}_i + b) < 0, \quad i : y_i = -1. \end{aligned}$$

Now, observe that if $(\mathbf{w}, b)$ is solution of problem (2.2.1), then $(\alpha\mathbf{w}, \alpha b)$ where $\alpha \neq 0$ is also solution of problem (2.2.1). Thus we impose the following constraint:

$$\min_{i=1,2,\ldots,m} \quad |\mathbf{w}^\top \mathbf{x}_i + b| = 1$$

Therefore, the problem (2.2.1) can be rewrite as follows

$$\begin{aligned} \max_{\mathbf{w},b} \quad & \frac{1}{\|\mathbf{w}\|}, \\ \text{subject to} \quad & \min_{i=1,2,\ldots,m} \quad |\mathbf{w}^\top \mathbf{x}_i + b| = 1 \\ & (\mathbf{w}^\top \mathbf{x}_i + b) > 0, \quad i : y_i = 1, \\ & (\mathbf{w}^\top \mathbf{x}_i + b) < 0, \quad i : y_i = -1, \end{aligned} \tag{2.2.2}$$

or equivalent

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}\|\mathbf{w}\|^2, \\ \text{subject to} \quad & (\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad i : y_i = 1, \\ & (\mathbf{w}^\top \mathbf{x}_i + b) \leq -1, \quad i : y_i = -1, \end{aligned} \tag{2.2.3}$$

Clearly, the problem (2.2.3) is convex optimization. Due to the objective function is convex and constraints are linear. The above convex optimization problem is called a support vector machine problem.

In practice, the support vector machine problem (2.2.3) may be hard to solve due to a training datasets cannot be linear separated. The way to apply the support vector machine problem to general cases of a training dataset is select an optimal hyperplane which maximum-margin and minimum number of error, i.e., we allow misclassification for some data points. This scheme can be expressed formally as follows

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \xi_i, \\ \text{subject to} \quad & y_i(\mathbf{x}_i^\top \mathbf{w} + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \quad i = 1, 2, 3, \ldots, m, \end{aligned} \tag{2.2.4}$$

where, $\mathbf{w} \in \mathbb{R}^n$ and $b \in \mathbb{R}$ are the weight vector and bias, respectively, which define the hyperplane $f(\mathbf{x}) : \mathbf{w}^\top \mathbf{x} + b = 0$, $C > 0$ is a constant parameter and $\boldsymbol{\xi} \in \mathbb{R}^m$ is slack

vectors. The decision function of the above formulation is based on the sign of $\mathbf{w}^\top \mathbf{x} + b$ where $\mathbf{x}$ is assigned to class $+1$ if the value is positive otherwise it is assigned to class -1. The basic idea of linear support vector classification is finding the solution to the primal problem (2.2.4) by means of solving its dual problem. The problem is

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i^\top \mathbf{x}_j) + \sum_{i=1}^{m} \alpha_i \\ \text{subject to} \quad & \sum_{i=1}^{m} \alpha_i y_i = 0, \\ & 0 \le \alpha_i \le C, \end{aligned} \tag{2.2.5}$$

The solutions of problem (2.2.4) is obtained by,

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i \quad \text{and} \quad b = y_j - \sum_{i=1}^{m} \alpha_i y_i (\mathbf{x}_i^\top \mathbf{x}_j),$$

Besides the support vector machine problem, the kernel methods are probably the best known technique that can be to extend a support vector machine problem to a linear separation in that hight dimensional space. Due to linear classification is often not possible in some cases.

Figure (3) is illustrates to use a non-linear mapping Φ from the input space X to a higher-dimensional Hilbert space H, where linear separation is possible. A solution to this problem is to use kernel method, whics is based on kernel function.

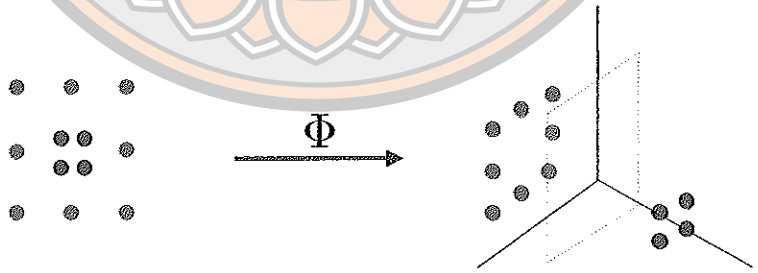


Figure 3: Illustration of kernel mapping

Definition 2.2.2. A function $K : X \times X \to \mathbb{R}$ is called a kernel over X, if for any $\mathbf{x}, \bar{\mathbf{x}} \in X$,

$$K(\mathbf{x}, \bar{\mathbf{x}}) = \langle \Phi(\mathbf{x}), \Phi(\bar{\mathbf{x}}) \rangle,$$

for some function Φ from X to Hilbert space H called a feature space.

The following example gives kernel function commonly used in application.

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

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

We now turn to the support vector machine problem, we can extend nonlinear classifiers, the kernel-generated surface is considered instead of hyperplane and is given by $\mathbf{u}^\top K(\mathbf{x}^\top, X^\top) + r = 0$ where $\mathbf{x}$ and $\mathbf{u}$ are column vector in $\mathbb{R}^n$, $r \in \mathbb{R}$, and $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m]^\top \in \mathbb{R}^{m \times n}$. The dual problem of nonlinear support vector machine problem is

$$\begin{aligned} \max_{\alpha} \quad & -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i=1}^{m} \alpha_i \\ \text{subject to} \quad & \sum_{i=1}^{m} \alpha_i y_i = 0, \\ & 0 \le \alpha \le C, \quad i = 1, 2, \ldots, m. \end{aligned} \tag{2.2.6}$$

In SVM model, we attempt to find a separating hyperplane $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b = 0, \mathbf{w} \in \mathbb{R}^n$, $b \in \mathbb{R}$; so as to maximize the margin and minimize the training error. Here, the training error is measured by the hinge loss function defined as

$$L_{hinge}(1 - y_i(\mathbf{x}_i^\top \mathbf{w} + b)) = \max\left\{0, 1 - y_i(\mathbf{x}_i^\top \mathbf{w} + b)\right\}. \tag{2.2.7}$$

This loss function is related to the shortest distance between the sets and the corresponding classifier leads to its sensitivity of noise and instability for resampling. In fact, the SVM problem (2.2.4) can rewrited into unconstrained optimization problem as follows

$$\min_{\mathbf{w}, b} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} L_{hinge}(1 - y_i(\mathbf{x}_i^\top \mathbf{w} + b)), \tag{2.2.8}$$

where $C > 0$ is a constant parameter.

Corollary 2.2.4. *[28, P. 207]: Problem* (2.2.4) *and problem* (2.2.8) *are equivalent.*

Proof. Fix some $\mathbf{w}, b$ and consider the minimization over $\boldsymbol{\xi}$ in problem (2.2.4).
Fix some i, since $\xi_i \ge 0$, the best assignment to ξ_i would be 0 if $y_i(x_i^\top \mathbf{w} + b) \ge 1$ and

would be $1 - y_i(\mathbf{x}_i^\top \mathbf{w} + b)$ otherwise.
In other words,

$$\xi_i = L_{hinge}(1 - y_i(\mathbf{x}_i^\top \mathbf{w} + b)) \quad \forall i.$$

Thus, this implies that Problem (2.2.4) and problem (2.2.8) are equivalent. □

However, to remediate this coming of a hinge loss function, Huang [26] suggests using a pinball loss function to the SVM classifier (Pin-SVM). This approach brings noise insensitivity. The way this model works by penalizing correctly classified samples, which is evident by the pinball loss function which is defined by

$$L_\tau(u) = \begin{cases} u, & \text{if } u \geq 0, \\ -\tau u, & \text{if } u < 0, \end{cases} \tag{2.2.9}$$

where $u = 1 - y_i(\mathbf{x}_i^\top \mathbf{w} + b)$ and $\tau \geq 0$ is a user-defined parameter. Then, the problem of support vector machin using pinball loss function is as follows:

$$\min_{\mathbf{w},b} \ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} L_\tau(1 - y_i(\mathbf{x}_i^\top \mathbf{w} + b)). \tag{2.2.10}$$

This problem is equivalently transformed into the constrained problem as follows:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \\ & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \leq 1 + \frac{1}{\tau}\xi_i, \\ & \xi_i \geq 0, \quad i = 1, 2, 3, \ldots, m. \end{aligned} \tag{2.2.11}$$

To write the dual problem of (2.2.11), we consider the following Lagrange function.

$$\begin{aligned} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \xi_i - \sum_{i=1}^{m} \boldsymbol{\alpha}_i(y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 + \xi_i) \\ - \sum_{i=1}^{m} \boldsymbol{\beta}_i(-y_i(\mathbf{w}^\top \mathbf{x}_i + b) + 1 + \frac{1}{\tau}\xi_i), \end{aligned}$$

where $\boldsymbol{\alpha} \geq \mathbf{0}, \boldsymbol{\beta} \geq \mathbf{0}$ and $C \geq 0$ are the Lagrange multipliers. The Karush–Kuhn–Tucker (KKT) necessary and sufficient optimality conditions for (2.2.11) are given by

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{m} (\alpha_i - \beta_i) y_i \mathbf{x}_i = 0, \tag{2.2.12}$$

$$\frac{\partial \mathcal{L}}{\partial b} = -\sum_{i=1}^{m}(\alpha_i - \beta_i)y_i = 0, \tag{2.2.13}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\xi}} = C - \alpha_i - \frac{1}{\tau}\beta_i = 0, \tag{2.2.14}$$

$$\alpha_i(-y_i(\mathbf{w}^\top \mathbf{x}_i + b) + 1 - \boldsymbol{\xi}_i) = 0, \tag{2.2.15}$$

$$\beta_i(y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 - \frac{1}{\tau}\boldsymbol{\xi}_i) = 0, \tag{2.2.16}$$

$$C\xi_i = 0, \tag{2.2.17}$$

where $i = 1, 2, 3, \ldots, m$. Thus, the dual problem of the support vector machine using pinball loss function is obtained as follows:

$$\begin{aligned} \max_{\boldsymbol{\lambda},\boldsymbol{\beta}} \quad & -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\lambda_i y_i(\mathbf{x}_i \cdot \mathbf{x}_j)y_j\lambda_j + \sum_{i=1}^{m}\lambda_i \\ \text{subject to} \quad & \sum_{i=1}^{m}\lambda_i y_i = 0 \\ & \lambda_i + (1 + \frac{1}{\tau})\beta_i = C, \quad i = 1, 2, \ldots, m, \\ & \lambda_i + \beta_i \geq 0,\ \beta_i > 0, \quad i = 1, 2, \ldots, m. \end{aligned} \tag{2.2.18}$$

Though the support vector machine using a pinball loss function is claimed more noise-insensitive than the support vector machine using a hinge loss function, in the process, it still loses sparsity. Based on the complementary slackness conditions and the constraint of dual problem (2.2.18), we know that

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) > 1 + \xi_i \quad \Rightarrow \quad \alpha_i = 0,\ \beta_i = \tau C,$$

and

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) < 1 - \frac{1}{\tau}\xi_i \quad \Rightarrow \quad \alpha_i = C,\ \beta_i = 0.$$

Note $\boldsymbol{\lambda} = \boldsymbol{\alpha} - \boldsymbol{\beta}$. Then, $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_i, \ldots, \lambda_m)$, we have $\lambda_i \neq 0$ for most $i.$, which means that the support vector machine using a pinball loss function loses the sparsity. In order to add this problem, in the same paper [26] proposed an ϵ-insensitive zone to Pin-SVM (ϵ-insensitive zone pinball support vector machine). Definition of pinball

loss with ϵ-insensitive zone is given by

$$L_\tau^\epsilon(u) = \begin{cases} u - \epsilon, & \text{if } u > \epsilon, \\ 0, & \text{if } -\frac{\epsilon}{\tau} \leq u \leq \epsilon, \\ -\tau(u + \frac{\epsilon}{\tau}), & \text{if } u < -\frac{\epsilon}{\tau}, \end{cases} \tag{2.2.19}$$

where $u = 1 - y_i(\mathbf{x}_i^\top \mathbf{w} + b)$ and $\epsilon \geq 0$ and $\tau \geq 0$ are user-defined parameters. Then any point located in the ϵ-insensitive zone (i.e., $[\tau, \epsilon]$) corresponds to a zero dual variable, providing sparsity to the model.

However, the ϵ-insensitive zone pinball support vector machine formulation requires an optimal choice of ϵ parameter needs to be prescribed. Therefore, Reshma [27] proposed the ϵ-insensitive zone Pin-support vector machine idea to develop the generalized pinball loss support vector machine. This technique allows asymmetric insensitive zone by allowing ϵ to be different.

The generalized pinball loss function is a generalization of other exiting loss functions which handle the problems of noise insensitivity and instability of re-sampling as well. The definition of generalized pinball loss function is given by

$$L_{\tau_1,\tau_2}^{\epsilon_1,\epsilon_2}(u) = \begin{cases} \tau_1(u - \frac{\epsilon_1}{\tau_1}), & \text{if } u > \frac{\epsilon_1}{\tau_1}, \\ 0, & \text{if } -\frac{\epsilon_2}{\tau_2} \leq u \leq \frac{\epsilon_1}{\tau_1}, \\ -\tau_2(u + \frac{\epsilon_2}{\tau_2}), & \text{if } u < -\frac{\epsilon_2}{\tau_2}, \end{cases} \tag{2.2.20}$$

where $u = 1 - y_i(\mathbf{x}_i^\top \mathbf{w} + b)$ and $\tau_1, \tau_2, \epsilon_1, \epsilon_2 \geq 0$ are user-defined parameters. With a generalized pinball loss function, the resulting formulation of SVM, leads to the following unconstrained optimization problem:

$$\min_{\mathbf{w},b} \ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} L_{\tau_1,\tau_2}^{\epsilon_1,\epsilon_2}(1 - y_i(\mathbf{x}_i^\top \mathbf{w} + b)), \tag{2.2.21}$$

where $L_{\tau_1,\tau_2}^{\epsilon_1,\epsilon_2}(\cdot)$ is a generalized pinball loss function. However, The proposed methods preserves the elegance of the classical SVM: the only difference in form is that different losses are used.

2.2.2 Twin Support Vector Machine (TSVM)

Let us consider a set of training data samples X in which m_1 data samples that belong to class $+1$ are represented by matrix $A \in \mathbb{R}^{m_1 \times n}$ while m_2 data samples

that belong to class -1 are represented by matrix $B \in \mathbb{R}^{m_2 \times n}$. Therefore, the sizes of matrices X is $(m_1 + m_2 \times n)$, where n is the dimension of feature space. To reduce the computational complexity of support vector machine, Jayadeva et al.[10] proposed the twin support vector machine (TSVM) based on hinge loss function. The formulation of TSVM solves two smaller QPPs. The TSVM classifier aims to determine two nonparallel planes:

$$f_1(\mathbf{x}) : \mathbf{w}_1^\top \mathbf{x} + b_1 = 0 \text{ and } f_2(\mathbf{x}) : \mathbf{w}_2^\top \mathbf{x} + b_2 = 0, \tag{2.2.22}$$

where, $\mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^n$ and $b_1, b_2 \in \mathbb{R}$ are the weight vectors and biases of the hyperplane $f_1(\mathbf{x})$ and the hyperplane $f_2(\mathbf{x})$, respectively. To find the pair of nonparallel hyperplanes, it is necessary to obtain solutions to constrained optimization problems:

$$\begin{aligned} \min_{\mathbf{w}_1, b_1, \boldsymbol{\xi}_2} \quad & \frac{1}{2}\|A\mathbf{w}_1 + \mathbf{e}_1 b_1\|^2 + c_1 \mathbf{e}_2^\top \boldsymbol{\xi}_2, \\ \text{subject to} \quad & -(B\mathbf{w}_1 + \mathbf{e}_2 b_1) + \boldsymbol{\xi}_2 \geq \mathbf{e}_2, \\ & \boldsymbol{\xi}_2 \geq 0, \end{aligned} \tag{2.2.23}$$

and

$$\begin{aligned} \min_{\mathbf{w}_2, b_2, \boldsymbol{\xi}_1} \quad & \frac{1}{2}\|B\mathbf{w}_2 + \mathbf{e}_2 b_2\|^2 + c_2 \mathbf{e}_1^\top \boldsymbol{\xi}_1, \\ \text{subject to} \quad & (A\mathbf{w}_2 + \mathbf{e}_1 b_2) + \boldsymbol{\xi}_1 \geq \mathbf{e}_1, \\ & \boldsymbol{\xi}_1 \geq 0, \end{aligned} \tag{2.2.24}$$

where $c_1, c_2 > 0$ are the penalty parameters, and $\boldsymbol{\xi}_1 \in \mathbb{R}^{m_1}$, $\boldsymbol{\xi}_2 \in \mathbb{R}^{m_2}$ are slack vectors.

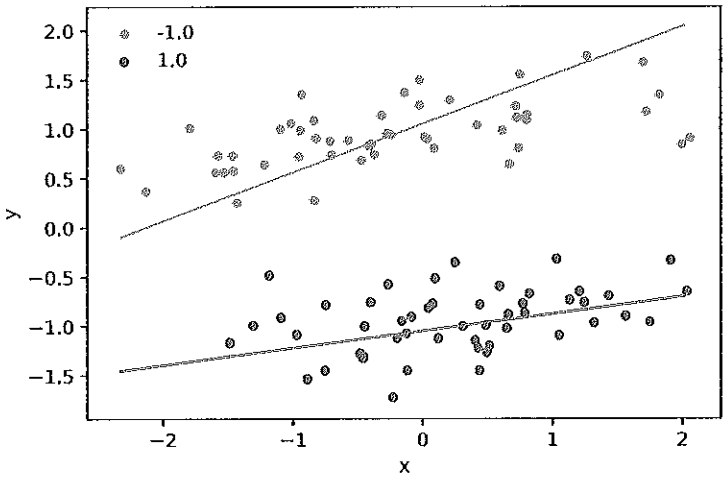


Figure 4: Illustration of the twin Support Vector Machine With Hinge Loss.

2.2.3 Twin parametric-margin support vector machine (TPSVM)

The classical SVM and TSVM assume that the probability distribution of training data is distributed uniformly. However, this probability distribution assumption is not always satisfied. To resolve this problem, the concept of TPSVM [16] was developed. The process of TPSVM is generating two nonparallel hyperplanes similar to the TSVM. Each hyperplane determines one of the parametric-margin hyperplanes. Note that $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ (defined in (3.4.3)) are positive and negative parametric-margin hyperplanes, respectively. Therefore, the TPSVM separates data if and only if:

$$\mathbf{w}_1^\top \mathbf{x}_i + b_1 \geq 0 \qquad \forall i = 1, 2, 3, ..., m_1, \tag{2.2.25}$$

$$\mathbf{w}_2^\top \mathbf{x}_i + b_2 \leq 0 \qquad \forall i = 1, 2, 3, ..., m_2. \tag{2.2.26}$$

In order to obtain the positive and negative parametric-margin hyperplanes, let us consider the following pair of constrained optimization problems:

$$\begin{aligned} \min_{\mathbf{w}_1, b_1, \boldsymbol{\xi}_1} \quad & \frac{1}{2}\|\mathbf{w}_1\|^2 + \frac{\nu_1}{m_2}\mathbf{e}_2^\top\Big(B\mathbf{w}_1 + \mathbf{e}_2 b_1\Big) + \frac{c_1}{m_1}\mathbf{e}_1^\top \boldsymbol{\xi}_1, \\ \text{subject to} \quad & A\mathbf{w}_1 + b_1\mathbf{e}_1 \geq 0 - \boldsymbol{\xi}_1, \\ & \boldsymbol{\xi}_1 \geq \mathbf{0}, \end{aligned} \tag{2.2.27}$$

and

$$\begin{aligned} \min_{\mathbf{w}_2, b_2, \boldsymbol{\xi}_2} \quad & \frac{1}{2}\|\mathbf{w}_2\|^2 + \frac{\nu_2}{m_1}\mathbf{e}_1^\top\Big(A\mathbf{w}_2 + \mathbf{e}_1 b_2\Big) + \frac{c_2}{m_2}\mathbf{e}_1^\top \boldsymbol{\xi}_2, \\ \text{subject to} \quad & B\mathbf{w}_2 + b_2\mathbf{e}_2 \leq 0 + \boldsymbol{\xi}_2, \\ & \boldsymbol{\xi}_2 \geq 0, \end{aligned} \tag{2.2.28}$$

where $c_1, c_2, \nu_1, \nu_2 > 0$ are the penalty parameters, and $\boldsymbol{\xi}_1, \boldsymbol{\xi}_2$ are slack vectors.

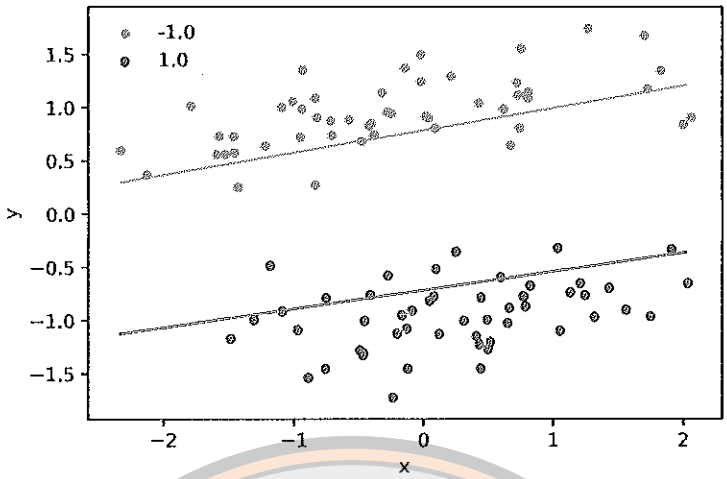


Figure 5: Illustration of the twin parametric-margin support vector machine with hinge loss.

However, it is worth noting that the TPSVM based on hinge loss function needs to be solved via quadratic programming problems (QPPs), which might lead to its sensitivity to noise, instability for re-sampling and its computational cost could leave a lot for large scale problems.

2.3 Stochastic Gradient Descent (SGD)

In this section is to describe the stochastic gradient descent method. We first briefly review a gradient method for minimizing a function $f(\mathbf{w})$. Let us consider a function $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable. The idea of this method is to start at $\mathbf{w}^{(k)}$ and move in the direction of the negative of $\eta_k \nabla f(\mathbf{w}^{(k)})$, where η_k is a positive scalar called the step size. The procedure leads to the following iterative algorithm:

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \eta_k \nabla f(\mathbf{w}^k).$$

The gradient method is to requires that the function f be differentiable. For the function f is nondiferentiable. The gradient method using a subgradient of f at $\mathbf{w}^{(k)}$, instead of the gradient.

For the stochastic gradient descent we do not require the update direction to be based exactly on the gradient. Instead, we allow the direction to be a random vector and only require that its expected value at each iteration will equal the gradient

direction. Or, more generally, we require that the expected value of the random vector will be a subgradient of the function at the current vector.

To make the stochastic gradient descent method easier to understand, we shall consider its applicability to learning tasks. Let us consider an unconstrained optimization problem (2.2.8). Note a training dataset is defined as $D = \{(\mathbf{x}_i, y_i) \in \mathbb{R}^n \times \{1, -1\} \,|\, i = 1, 2, 3, \ldots, m\}$, $\mathbf{w} = [\mathbf{w}^\top, b]^\top$ where $\mathbf{w}$ is column vectors in $\mathbb{R}^{n+1}$, and $\mathbf{x} = [\mathbf{x}^\top, 1]^\top$. On each iteration stochastic gradient descent as follow. Initially $\mathbf{w}^{(1)}$ and positive parameters η, T. For $t = 1, \ldots, T$, on iteration t of the algorithm, we choose a random training sample $(\mathbf{x}_{i_t}, y_{i_t})$ by picking an index $i_t \in \{i \,|\, i = 1, \ldots, m\}$. We then replace the objective of (2.2.8) with an approximation based on the training example $(\mathbf{x}_{i_t}, y_{i_t})$, yielding:

$$f_{i_t}(\mathbf{w}) := \frac{1}{2}\|\mathbf{w}\|^2 + CL_{hinge}(1 - y_{i_t}(\mathbf{x}_{i_t}^\top \mathbf{w})).$$

Then consider the sub-gradient of the above approximate objective and update

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla_{\mathbf{w}} f_{i_t}(\mathbf{w}).$$

Eventually, after T iterations, the stochastic gradient descent algorithm outputs the averaged vector $\bar{\mathbf{w}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}^{(t)}$.

Recently, Wang [25] rely on the stochastic gradient descent framework for solving twin support vector machine problem. He has shown that twin support vector machine problem by using a stochastic gradient descent method can handle large scale problems efficiently. However, it still that based on a hinge loss function may lead to its sensitivity to the noise and instability for re-sampling.

Therefore, in view of the stochastic gradient descent-based optimization in the twin support vector machine problem [25] and motivated with generalized pinball loss SVM and twin parametric-margin support vector machine, we formulate a twin parametric support vector machine model which is based on generalized pinball loss function by using an iterative method such as stochastic gradient descent (SG-TPSVM).

CHAPTER III

MAIN RESULTS

In this chapter, we shall first formulate linear twin parametric support vector machine model which is based on generalized pinball loss function by using a stochastic gradient method. We then introduce the idea of kernels to formulate non-linear twin parametric support vector machine model which is based on generalized pinball loss function by using a stochastic gradient method. Finally, we turn to analyze the convergence of the proposed model.

3.4 Stochastic Gradient method linear twin parametric-margin support vector machine with generalized pinball loss function (linear SG-TPSVM)

Following the method of formulating the TPSVM problems (discussed in (2.2.27) and (2.2.28)), we incorporate the generalized pinball loss function in the objective function to get the convex unconstrained minimization problems as follows:

$$\min_{\omega_1} \frac{1}{2}\|\omega_1\|^2 + \frac{\nu_1}{m_2}\sum_{j=1}^{m_2}\left(\omega_1^\top \mathbf{x}_j^-\right) + \frac{c_1}{m_1}\sum_{i=1}^{m_1} L_{\tau_1,\tau_2}^{\epsilon_1,\epsilon_2}\left(0 - y_i^+(\omega_1^\top \mathbf{x}_i^+)\right), \tag{3.4.1}$$

$$\min_{\omega_2} \frac{1}{2}\|\omega_2\|^2 - \frac{\nu_2}{m_1}\sum_{i=1}^{m_1}\left(\omega_2^\top \mathbf{x}_i^+\right) + \frac{c_2}{m_2}\sum_{j=1}^{m_2} L_{\tau_3,\tau_4}^{\epsilon_3,\epsilon_4}\left(0 - y_j^-(\omega_2^\top \mathbf{x}_j^-)\right), \tag{3.4.2}$$

where $\omega_1 = [\mathbf{w}_1^\top, b_1]^\top$, $\omega_2 = [\mathbf{w}_2^\top, b_2]^\top$, $\mathbf{x} = [\mathbf{x}^\top, 1]^\top$ and $\mathbf{w}_1, \mathbf{w}_2$ are column vectors in $\mathbb{R}^n$, $b_1, b_2 \in \mathbb{R}$, $y^+ = 1$, $y^- = -1$, and $c_1, c_2, \nu_1, \nu_2 > 0$ are positive parameters. For a linear case, the SG-TPSVM model finds two hyperplanes in $\mathbb{R}^n$ as follows:

$$f_1(\mathbf{x}) : \omega_1^\top \mathbf{x} = 0 \quad \text{and} \quad f_2(\mathbf{x}) : \omega_2^\top \mathbf{x} = 0. \tag{3.4.3}$$

Let us consider $t \geq 1$ be an iteration count. To solve the above two problems, we construct a series of function $f_{1,t}$ and $f_{2,t}$ as follows:

$$f_{1,t}(\omega_1) := \frac{1}{2}\|\omega_1\|^2 + \nu_1\left(\omega_1^\top \mathbf{x}_t^-\right) + c_1 L_{\tau_1,\tau_2}^{\epsilon_1,\epsilon_2}\left(0 - y_t^+(\omega_1^\top \mathbf{x}_t^+)\right), \tag{3.4.4}$$

$$f_{2,t}(\omega_2) := \frac{1}{2}\|\omega_2\|^2 - \nu_2\left(\omega_2^\top \mathbf{x}_t^+\right) + c_2 L_{\tau_3,\tau_4}^{\epsilon_3,\epsilon_4}\left(0 - y_t^-(\omega_2^\top \mathbf{x}_t^-)\right), \tag{3.4.5}$$

where $\mathbf{x}_t^+$ and $\mathbf{x}_t^-$ are selected randomly from A and B, respectively.

We now define a subgradient of the above function at $\boldsymbol{\omega}_{1,t}$ and $\boldsymbol{\omega}_{2,t}$. Let $\mathbf{s}_t$ and $\tilde{\mathbf{s}}_t$ be a subgradients of $f_{1,t}$ and $f_{2,t}$ associated with the samples $\mathbf{x}_t^+$ and $\mathbf{x}_t^-$ at point $\boldsymbol{\omega}_{1,t}$ and point $\boldsymbol{\omega}_{2,t}$, respectively. That is,

$$\mathbf{s}_t \in \partial f_{1,t}(\boldsymbol{\omega}_{1,t}) = \boldsymbol{\omega}_{1,t} + \nu_1 \mathbf{x}_t^- - c_1 \partial L_{\tau_1,\tau_2}^{\epsilon_1,\epsilon_2}(0 - y_t^+(\boldsymbol{\omega}_{1,t}^\top \mathbf{x}_t^+)) y_t^+(\mathbf{x}_t^+), \tag{3.4.6}$$

$$\tilde{\mathbf{s}}_t \in \partial f_{2,t}(\boldsymbol{\omega}_{2,t}) = \boldsymbol{\omega}_{2,t} - \nu_2 \mathbf{x}_t^+ - c_2 \partial L_{\tau_1,\tau_2}^{\epsilon_1,\epsilon_2}(0 - y_t^-(\boldsymbol{\omega}_{2,t}^\top \mathbf{x}_t^-)) y_t^-(\mathbf{x}_t^-), \tag{3.4.7}$$

where

$$\partial L_{\tau_1,\tau_2}^{\epsilon_1,\epsilon_2}(u) = \begin{cases} \{-\tau_1\} & \text{if } u > \frac{\epsilon_1}{\tau_1}, \\ [-\tau_1, 0] & \text{if } u = \frac{\epsilon_1}{\tau_1}, \\ \{0\} & \text{if } -\frac{\epsilon_2}{\tau_2} < u < \frac{\epsilon_1}{\tau_1}, \\ [0, \tau_2] & \text{if } u = -\frac{\epsilon_2}{\tau_2}, \\ \{\tau_2\} & \text{if } u < -\frac{\epsilon_2}{\tau_2}. \end{cases} \tag{3.4.8}$$

With the above notations and the existence of $l_t \in \partial L_{\tau_1,\tau_2}^{\epsilon_1,\epsilon_2}(0 - y_t^+(\boldsymbol{\omega}_{1,t}^\top \mathbf{x}_t^+))$ and $\tilde{l}_t \in \partial L_{\tau_1,\tau_2}^{\epsilon_1,\epsilon_2}(0 - y_t^-(\boldsymbol{\omega}_{2,t}^\top \mathbf{x}_t^-))$, sudgradients $\mathbf{s}_t$ and $\tilde{\mathbf{s}}_t$ can be rewritten as:

$$\mathbf{s}_t = \boldsymbol{\omega}_{1,t} + \nu_1 \mathbf{x}_t^- - c_1 l_t y_t^+(\mathbf{x}_t^+), \tag{3.4.9}$$

$$\tilde{\mathbf{s}}_t = \boldsymbol{\omega}_{2,t} - \nu_2 \mathbf{x}_t^+ - c_2 \tilde{l}_t y_t^-(\mathbf{x}_t^-). \tag{3.4.10}$$

The proposed SG-TPSVM starts from the initial $w_{1,t}$ and $w_{2,t}$. Then, the updates are as follows:

$$\boldsymbol{\omega}_{1,t+1} = \boldsymbol{\omega}_{1,t} - \eta_t \mathbf{s}_t, \tag{3.4.11}$$

$$\boldsymbol{\omega}_{2,t+1} = \boldsymbol{\omega}_{2,t} - \eta_t \tilde{\mathbf{s}}_t. \tag{3.4.12}$$

A new data sample $\mathbf{x} \in \mathbb{R}^n$ is assigned to class $y\,(y = -1, 1)$ depending on which of the two planes given by (3.4.3) it lies closest to, i.e.,

$$\bar{\mathrm{y}} = \arg\min_{i=1,2} \frac{|\bar{\mathbf{x}}^\top \mathbf{w}_i + b_i|}{\|\mathbf{w}_i\|}, \tag{3.4.13}$$

where $|.|$ denotes the absolute value and, hence, the procedure for training a SG-TPSVM model is summarized in Algorithm 1.

Algorithm 1 SG-TPSVM

Input: Positive class $A \in \mathbb{R}^{m_1 \times n}$, negative class $B \in \mathbb{R}^{m_2 \times n}$,
positive parameters c_1, c_2, ν_1, ν_2 and tolerance: $tol = 10^{-4}$

Set $\omega_{1,1}$ and $\omega_{2,1}$ to be zero;
while $\|\mathbf{s}_t\| \geq tol$ **do**
    Choose a pair of samples $\mathbf{x}_t^+$ and $\mathbf{x}_t^-$ at random from A and B, respectively.
    Compute stochastic gradient $\mathbf{s}_t$ using Eqs. (3.4.9)
    Update $\omega_{1,t+1}$ using Eqs. (3.4.11)
    $t = t + 1$
end
while $\|\tilde{\mathbf{s}}_t\| \geq tol$ **do**
    Choose a pair of samples $\mathbf{x}_t^+$ and $\mathbf{x}_t^-$ at random from A and B, respectively.
    Compute stochastic gradient $\tilde{\mathbf{s}}_t$ using Eqs. (3.4.10)
    Update $\omega_{2,t+1}$ using Eqs. (3.4.12)
    $t = t + 1$
end

Output: $\omega_1^* = \frac{1}{T}\sum_{t=1}^{T} \omega_{1,t}$ and $\omega_2^* = \frac{1}{T}\sum_{t=1}^{T} \omega_{2,t}$

3.5 Stochastic Gradient method non-linear twin parametric-margin support vector machine with generalized pinball loss function (nonlinear SG-TPSVM)

By employing the kernel method, we extend our SG-TPSVM to the non-linear case. Suppose that $K(\cdot,\cdot)$ is the predefined kernel function. For a nonlinear case, the SG-TPSVM model finds two surfaces as the following:

$$f_1(\mathbf{x}) : \omega_1^\top K(\mathbf{x}, X) = 0 \quad \text{and} \quad f_2(\mathbf{x}) : \omega_2^\top K(\mathbf{x}, X) = 0, \tag{3.5.1}$$

where $\omega_1 = [\mathbf{w}_1^\top, b_1]^\top$, $\omega_2 = [\mathbf{w}_2^\top, b_2]^\top$, $\mathbf{x} = [\mathbf{x}^\top, 1]^\top$ and $\mathbf{w}_1, \mathbf{w}_2$ are column vectors in $\mathbb{R}^{m_1+m_2}$, $b_1, b_2 \in \mathbb{R}$, $X = \begin{bmatrix} A_{m_1 \times n} \\ B_{m_2 \times n} \end{bmatrix}$ and $K(\mathbf{x}, X) = [K(\mathbf{x}^\top, X^\top), 1]^\top$ are column

augmented matrices. The kernel counterparts of (3.4.1) and (3.4.2) can be formulated as

$$\min_{\omega_1} \ \frac{1}{2}\|\omega_1\|^2 + \frac{\nu_1}{m_2}\sum_{j=1}^{m_2}\Big(\omega_1^\top K(\mathbf{x}_j^-, X)\Big) + \frac{c_1}{m_1}\sum_{i=1}^{m_1} L_{\tau_1,\tau_2}^{\epsilon_1,\epsilon_2}\Big(0 - y_i^+(\omega_1^\top K(\mathbf{x}_i^+, X))\Big), \tag{3.5.2}$$

$$\min_{\omega_2} \ \frac{1}{2}\|\omega_2\|^2 - \frac{\nu_2}{m_1}\sum_{i=1}^{m_1}\Big(\omega_2^\top K(\mathbf{x}_i^+, X)\Big) + \frac{c_2}{m_2}\sum_{j=1}^{m_2} L_{\tau_3,\tau_4}^{\epsilon_3,\epsilon_4}\Big(0 - y_j^-(\omega_2^\top K(\mathbf{x}_j^-, X))\Big), \tag{3.5.3}$$

where $c_1, c_2, \nu_1, \nu_2 > 0$ are the positive parameters.

The solution to the above-mentioned optimization problems (3.5.2) and (3.5.3) can be computed similar to the linear case using Algorithm 1. Furthermore, a new data point $\mathbf{x} \in \mathbb{R}^n$ is assigned to class $y (y = -1, +1)$ according to the equation (3.4.13).

For large scale problems, it is time consuming to calculate the kernel $K(\cdot, X)$. However, the reduced kernel strategy [30], which has been successfully applied to SVM and TSVM [30, 25], can also be applied to our SG-TPSVM. This strategy replaces $K(\cdot, X)$ with $K(\cdot, \bar{X})$, where $\bar{X}$ is a randomly sampled subset of X

3.6 Convergence Analysis

In this section, we analyze the convergence of the proposed SG-TPSVM model. For convenience, we only consider the first problem (3.4.1) together with the SGD formulation of the linear SG-TPSVM. The conclusions for another problem (3.4.2) and the nonlinear algorithm can be obtained similarly. Let $\boldsymbol{\omega} = [\mathbf{w}_1^\top, b_1]^\top$, $A = [A, \mathbf{e}]$, $B = [B, \mathbf{e}]$ where A and B are samples of class $+1$ and -1, respectively, and $\mathbf{x} = [\mathbf{x}^\top, 1]^\top$. Then, the the first problem (3.4.1) is reformulated as

$$\min_{\omega} \ \ f(\omega) \ := \ \frac{1}{2}\|\omega\|^2 + \frac{\nu_1}{m_2}\sum_{j=1}^{m_2}\Big(\omega^\top \mathbf{x}_j^-\Big) + \frac{c_1}{m_1}\sum_{i=1}^{m_1} L_{\tau_1,\tau_2}^{\epsilon_1,\epsilon_2}\Big(0 - y_i^+(\omega^\top \mathbf{x}_i^+)\Big), \tag{3.6.1}$$

Next, we reformulate the t-th $(t \geq 1)$ function in SG-TPSVM as

$$f_t(\omega) := \frac{1}{2}\|\omega\|^2 + \nu_1\Big(\omega^\top \mathbf{x}_t^-\Big) + c_1 L_{\tau_1,\tau_2}^{\epsilon_1,\epsilon_2}\Big(0 - y_t^+(\omega^\top \mathbf{x}_t^+)\Big), \tag{3.6.2}$$

where $\mathbf{x}_t^+$ and $\mathbf{x}_t^-$ are the selected uniformly random from $\bar{A}$ and $\bar{B}$, respectively. For t-th iteration, the subgradient of f_t at $\boldsymbol{\omega}_t$ is denote by

$$\mathbf{s}_t = \boldsymbol{\omega}_t + \nu_1 \mathbf{x}_t^- - c_1 l_t y_t^+(\mathbf{x}_t^+), \tag{3.6.3}$$

Given $\boldsymbol{\omega}_1$ and the step size $\eta_t = \frac{1}{t}$, $\boldsymbol{\omega}_{t+1}$ for $t \geq 1$ is updated by

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t - \eta_t \mathbf{s}_t, \tag{3.6.4}$$

i.e.,

$$\begin{aligned}\boldsymbol{\omega}_{t+1} &= \boldsymbol{\omega}_t - \eta_t \mathbf{s}_t \\ &= \boldsymbol{\omega}_t - \frac{1}{t}\Big(\boldsymbol{\omega}_t + \nu_1 \mathbf{x}_t^- - c_1 l_t y_t^+(\mathbf{x}_t^+)\Big) \\ &= (1 - \frac{1}{t})\boldsymbol{\omega}_t - \frac{1}{t}\Big(\nu_1 \mathbf{x}_t^- - c_1 l_t y_t^+(\mathbf{x}_t^+)\Big).\end{aligned} \tag{3.6.5}$$

To prove the convergence of **Algorithm 1**, we use the notions of Convergence for Sequences of Random Variables. Consider a sequence of random variables $\{\boldsymbol{\omega}_t\}$ that is defined on an underlying sample space X. First, we rely on the following lemma.

Lemma 3.6.1. *The sequence $\{\|\boldsymbol{\omega}_t\|\}$ and the sequence $\{\|s_t\|\}$ have upper bounds, where $\boldsymbol{\omega}_t$ and s_t are defined by* (3.6.5) *and* (3.6.3)*, respectively.*

Proof. The formulation (3.6.5) can be rewritten as

$$\boldsymbol{\omega}_{t+1} = Z_t \boldsymbol{\omega}_t + \frac{1}{t}\mathbf{v}_t, \tag{3.6.6}$$

where $Z_t = \Big(\frac{t-1}{t}\Big) I$, I is the identity matrix, and $\mathbf{v}_t = -\nu_1 \mathbf{x}_t^- + c_1 l_t y_t(\mathbf{x}_t^+)$. For $t \geq 2$, Z_t is positive definite, and the largest eigenvalue λ_t of Z_t is equal to $\dfrac{t-1}{t}$. From (3.6.6), we obtrain that

$$\boldsymbol{\omega}_{t+1} = \prod_{i=2}^{t} Z_i \boldsymbol{\omega}_2 + \sum_{i=2}^{t} \bar{Z}_i, \tag{3.6.7}$$

where

$$\bar{Z}_i = \begin{cases} \frac{1}{i}\Big(\prod_{j=i+1}^{t} Z_j\Big)\mathbf{v}_i & \text{if } \ i < t, \\ \frac{1}{i}\mathbf{v}_i & \text{if } \ i = t. \end{cases} \tag{3.6.8}$$

For $i \geq 2$,

$$\|Z_i \boldsymbol{\omega}_2\| \ \leq \ \|Z_i\| \|\boldsymbol{\omega}_2\| \ = \ \lambda_i \|\boldsymbol{\omega}_2\| \ = \ \frac{i-1}{i}\|\boldsymbol{\omega}_2\|.$$

Therefore,

$$\left\| \prod_{i=2}^{t} Z_i \boldsymbol{\omega}_2 \right\| = \|Z_2 Z_3 \cdots Z_t \boldsymbol{\omega}_2\|$$

$$\begin{aligned}
&\leq \|Z_2\|\|Z_3\|\cdots\|Z_t\|\|\omega_2\| \\
&\leq \frac{(2-1)}{2}\cdot\frac{(3-1)}{3}\cdot\cdots\cdot\frac{(t-1)}{t}\|\omega_2\| \\
&\leq \frac{1}{t}\|\omega_2\|, \qquad (3.6.9)
\end{aligned}$$

Next, we consider $\|\bar{Z}_i\|$, case I, for $i < t$ we have

$$\begin{aligned}
\|\bar{Z}_i\| &= \left\|\frac{1}{i}\Big(\prod_{j=i+1}^{t} Z_j\Big)\mathbf{v}_i\right\| \\
&\leq \frac{1}{i}\|Z_{i+1}\|\|Z_{i+2}\|\cdots\|Z_t\|\|\mathbf{v}_i\| \\
&= \frac{1}{i}\cdot\frac{(i+1)-1}{i+1}\cdot\frac{(i+2)-1}{i+2}\cdots\frac{t-1}{t}\|\mathbf{v}_i\| \\
&= \frac{1}{t}\|\mathbf{v}_i\| \\
&\leq \frac{1}{t}\max_{i<t}\|\mathbf{v}_i\|,
\end{aligned}$$

and case II, for $i = t$ we have

$$\|\bar{Z}_i\| = \frac{1}{t}\|\mathbf{v}_t\| = \frac{1}{t}\max_{i=t}\|\mathbf{v}_i\|.$$

From case I and case II, we obtain that

$$\|\bar{Z}_i\| \leq \frac{1}{t}\max_{i\leq t}\|\mathbf{v}_i\|, \qquad \text{for } i \geq 2. \qquad (3.6.10)$$

Thus, we have

$$\begin{aligned}
\|\omega_{t+1}\| &= \left\|\prod_{i=2}^{t} Z_i\omega_2 + \sum_{i=2}^{t}\bar{Z}_i\right\| \\
&\leq \left\|\prod_{i=2}^{t} Z_i\omega_2\right\| + \sum_{i=2}^{t}\|\bar{Z}_i\| \\
&\leq \frac{1}{t}\|\omega_2\| + \sum_{i=2}^{t}\frac{1}{t}\max_{i\leq t}\|\mathbf{v}_i\| + \frac{1}{t}\max_{i\leq t}\|\mathbf{v}_i\| - \frac{1}{t}\max_{i\leq t}\|\mathbf{v}_i\| \\
&= \frac{1}{t}\|\omega_2\| + t\Big(\frac{1}{t}\max_{i\leq t}\|\mathbf{v}_i\|\Big) - \Big(\frac{1}{t}\max_{i\leq t}\|\mathbf{v}_i\|\Big) \\
&= \frac{1}{t}\|\omega_2\| + \frac{t-1}{t}\Big(\max_{i\leq t}\|\mathbf{v}_i\|\Big) \\
&\leq \|\omega_2\| + \nu_1\max_{\mathbf{x}\in B}\|\mathbf{x}\| + c_1\max\{\tau_1,\tau_2\}\max_{\mathbf{x}\in A}\|\mathbf{x}\|. \qquad (3.6.11)
\end{aligned}$$

Let M_1 and M_2 be the largest norms of the samples in the set A and set B, respectively, and

$$G_1 = \max\left\{\|\omega_1\|, \|\omega_2\| + \nu_1 M_2 + c_1\max\{\tau_1,\tau_2\}M_1\right\}.$$

Therefore G_1 is an upper bound of $\{\|\boldsymbol{\omega}_t\|\}$. Next, we consider, for any $t \geq 1$

$$\begin{aligned}\|\mathbf{s}_t\| &= \|\boldsymbol{\omega}_t + \nu_1 \mathbf{x}_t^- - c_1 l_t y_t(\mathbf{x}_t^+)\| \\ &\leq \|\boldsymbol{\omega}_t\| + \nu_1 \|\mathbf{x}_t^-\| + c_1 \max\{\tau_1, \tau_2\} \|\mathbf{x}_t^+\| \\ &\leq \|\boldsymbol{\omega}_t\| + \nu_1 M_2 + c_1 \max\{\tau_1, \tau_2\} M_1 \\ &\leq G_1 + \nu_1 M_2 + c_1 \max\{\tau_1, \tau_2\} M_1.\end{aligned}$$

We obtain that $G_2 = G_1 + \nu_1 M_2 + c_1 \max\{\tau_1, \tau_2\} M_1$ being an upper bound of $\{\|\mathbf{s}_t\|\}$. □

Now, we can establish a convergence of **Algorithm 1**.

Theorem 3.6.2. *The sequence $\{\boldsymbol{\omega}_t\}$ generated by Algorithm 1 almost surely converge to an optimal solution of problem* (3.6.1).

Proof. On the one hand, from (3.6.9) and Lemma 4.1, we have

$$\lim_{t\to\infty} \left\| \prod_{i=2}^{t} Z_i \boldsymbol{\omega}_2 \right\| = 0, \tag{3.6.12}$$

which implies that

$$\lim_{t\to\infty} \prod_{i=2}^{t} Z_i \boldsymbol{\omega}_2 = 0. \tag{3.6.13}$$

On the other hand, from (3.6.10), we have

$$\begin{aligned}\sum_{i=2}^{t} \|\bar{Z}_i\| &\leq \sum_{i=2}^{t} \left(\frac{1}{t} \max_{i\leq t} \|\mathbf{v}_i\| \right) \\ &\leq v_1 M_2 + c_1 \max\{\tau_1, \tau_2\} M_1. \end{aligned} \tag{3.6.14}$$

Since (3.6.14), we have $\sum_{i=2}^{t} \|\bar{Z}_i\|$ is bounded. And we know that $\sum_{i=2}^{t} \|\bar{Z}_i\|$ is monotone increasing, this implies that $\sum_{i=2}^{t} \|\bar{Z}_i\|$ is convergent. By infinite series of vectors is convergent if its norm series is convergent, we have $\sum_{i=2}^{t} \bar{Z}_i$ is convergent. Then, since

$$\boldsymbol{\omega}_{t+1} = \prod_{i=2}^{t} Z_i \boldsymbol{\omega}_2 + \sum_{i=2}^{t} \bar{Z}_i,$$

we have the sequence $\{\omega_t\}$ is convergent. Then, we assume that $\{\omega_t\}$ converges to ω^*, for some $\omega^* \in \mathbb{R}^n$. We will show that ω^* is an optimal solution of problem (3.6.1). Let us consider, for all $\omega \in \mathbb{R}^n$. Since $\mathbf{s}_t$ is a subgradient of f at ω_t, we have

$$\begin{aligned} f(\omega_t) - f(\omega) &\leq \langle \mathbf{s}_t, \omega_t - \omega \rangle \\ &= \langle \mathbf{s}_t, \omega_t - \omega + \omega^* - \omega^* \rangle \\ &= \langle \mathbf{s}_t, \omega_t - \omega^* \rangle + \langle \mathbf{s}_t, \omega^* - \omega \rangle \\ &\leq \|\mathbf{s}_t\| \|\omega_t - \omega^*\| + \|\mathbf{s}_t\| \|\omega^* - \omega\|. \end{aligned}$$

Taking $t \to \infty$, and by Lemma 3.3.1, we have

$$\lim_{t\to\infty} \Big(f(\omega_t) - f(\omega) \Big) \leq 0. \tag{3.6.15}$$

Since f is continuous and $\omega_t \to \omega^*$, this implies that

$$f(\omega^*) \leq f(\omega), \quad \forall \omega \in \mathbb{R}^n. \tag{3.6.16}$$

Thus, ω^* is an optimal solution of problem (3.6.1). Hence, we conclude that the sequence $\{\omega_t\}$ generated by Algorithm 1 almost surely converge to an optimal solution of problem (3.6.1). □

Theorem 3.3.2 says that the first of two iterative problems in the sequence generated by Algorithm 1 is convergent. The same conclusion can be obtained for the nonlinear case. Thus, we immediately have the sequences generated by Algorithm 1 convergent. Further, in order to establish a bound on the expected convergence rate of the proposed SG-TPSVM model, we need to justify the following lemma.

Lemma 3.6.3. *For each $t = 1, 2, \ldots, T$, assume that SGD is run for T iterations with $\eta_t = \frac{1}{t}$. For an algorithm with an initialization $\omega_1 = \mathbf{0}$ we have,*

$$\sum_{t=1}^{T} \langle \omega_t - \omega^*, s_t \rangle \quad \leq \quad TG_2(G_1 + \|\omega^*\|) + \frac{1}{2} G_2^2 (1 + \ln T), \tag{3.6.17}$$

where G_1, G_2 are an upper bound of $\{\|\omega_t\|\}$ and $\{\|\mathbf{s}_t\|\}$, respectively.

Proof. Consider,

$$\langle \omega_t - \omega^*, \mathbf{s}_t \rangle = \frac{1}{\eta_t} \langle \omega_t - \omega^*, \eta_t \mathbf{s}_t \rangle \tag{3.6.18}$$

$$= \frac{1}{2\eta_t}\Big(\|\omega_t - \omega^*\|^2 - \|\omega_{t+1} - \omega^*\|^2 + \eta_t^2\|\mathrm{s}_t\|^2\Big). \tag{3.6.19}$$

Summing up these equality over t and using Lemma 3.1, we obtain that

$$\begin{aligned}
\sum_{t=1}^{T}\langle \omega_t - \omega^*, \mathrm{s}_t\rangle &= \sum_{t=1}^{T}\Big(\frac{1}{2\eta_t}\Big(\|\omega_t - \omega^*\|^2 - \|\omega_{t+1} - \omega^*\|^2 + \eta_t^2\|\mathrm{s}_t\|^2\Big)\Big) \\
&= \frac{1}{2}\sum_{t=1}^{T}\frac{1}{\eta_t}\Big(\|\omega_t - \omega^*\|^2 - \|\omega_{t+1} - \omega^*\|^2\Big) \\
&\quad + \frac{1}{2}\sum_{t=1}^{T}\Big(\eta_t\|\mathrm{s}_t\|^2\Big) \\
&= \frac{1}{2}\Big(\sum_{t=1}^{T} t\|\omega_t - \omega^*\|^2 - \sum_{t=1}^{T} t\|\omega_{t+1} - \omega^*\|^2\Big) \\
&\quad + \frac{1}{2}\sum_{t=1}^{T}\Big(\eta_t\|\mathrm{s}_t\|^2\Big) \\
&= \frac{1}{2}\Big(\|\omega_1 - \omega^*\|^2 + 2\|u_2 - \omega^*\|^2 + \cdots + T\|\omega_T - \omega^*\|^2 \\
&\quad - \|\omega_2 - \omega^*\|^2 - 2\|\omega_3 - \omega^*\|^2 - \cdots \\
&\quad - (T-1)\|\omega_T - \omega^*\|^2 - T\|\omega_{T+1} - \omega^*\|^2\Big) \\
&\quad + \frac{1}{2}\sum_{t=1}^{T}\Big(\eta_t\|\mathrm{s}_t\|^2\Big) \\
&= \frac{1}{2}\Big(\|\omega_1 - \omega^*\|^2 + \|\omega_2 - \omega^*\|^2 + \cdots + \|\omega_T - \omega^*\|^2 \\
&\quad - T\|\omega_{T+1} - \omega^*\|^2\Big) + \frac{1}{2}\sum_{t=1}^{T}\Big(\eta_t\|\mathrm{s}_t\|^2\Big) \\
&= \frac{1}{2}\Big(\sum_{t=1}^{T}\|\omega_t - \omega^*\|^2 - T\|\omega_{T+1} - \omega^*\|^2\Big) \\
&\quad + \frac{1}{2}\sum_{t=1}^{T}\Big(\eta_t\|\mathrm{s}_t\|^2\Big) \\
&\leq \Big(G_1 + \|\omega^*\|\Big)\sum_{t=1}^{T}\|\omega_{T+1} - \omega_t\| + \frac{1}{2}G_2^2\Big(1 + \ln T\Big) \\
&= \Big(G_1 + \|\omega^*\|\Big)\sum_{t=1}^{T}\|\sum_{t=1}^{T}\frac{1}{t}\mathrm{s}_t\| + \frac{1}{2}G_2^2\Big(1 + \ln T\Big) \\
&\leq TG_2\Big(G_1 + \|\omega^*\|\Big) + \frac{1}{2}G_2^2\Big(1 + \ln T\Big).
\end{aligned} \tag{3.6.20}$$

Hence, our Lemma has been proved. □

Theorem 3.6.4. *Under the assumptions of Lemma 4.3, let* $\bar{\omega} = \frac{1}{T}\sum_{t=1}^{T}\omega_t$ *be the*

solution output of SG-TPSVM. Then

$$f(\bar{\omega}) - f(\omega^*) \quad \leq \quad G_2\Big(G_1 + \|\omega^*\|\Big) + \frac{1}{2T}G_2^2\Big(1 + \ln T\Big), \tag{3.6.21}$$

where G_1, G_2 *are an upper bound of* $\{\|\omega_t\|\}$ *and* $\{\|s_t\|\}$*, respectively.*

Proof. From the definition of $\bar{\omega}$, we have

$$\begin{aligned} f(\bar{\omega}) - f(\omega^*) &= f\Big(\frac{1}{T}\sum_{t=1}^{T}\omega_t\Big) - f(\omega^*) \\ &\leq \frac{1}{T}\sum_{t=1}^{T}\Big(f(\omega_t)\Big) - f(\omega^*) \\ &= \frac{1}{T}\sum_{t=1}^{T}\Big(f(\omega_t) - f(\omega^*)\Big). \end{aligned} \tag{3.6.22}$$

As f is a convex function and s_t is the subgradient of f at ω_t, we have

$$f(\omega_t) - f(\omega^*) \leq \langle \omega_t - \omega^*, \mathrm{s}_t \rangle. \tag{3.6.23}$$

Also, taking sum over t and divided by T, it follows that

$$\frac{1}{T}\sum_{t=1}^{T}\Big(f(\omega_t) - f(\omega^*)\Big) \leq \frac{1}{T}\sum_{t=1}^{T}\langle \omega_t - \omega^*, \mathrm{s}_t \rangle. \tag{3.6.24}$$

By Lemma 3.3.3, we have

$$\frac{1}{T}\sum_{t=1}^{T}\langle \omega_t - \omega^*, \mathrm{s}_t \rangle \leq G_2(G_1 + \|\omega^*\|) + \frac{1}{2T}G_2^2(1 + \ln T). \tag{3.6.25}$$

Combining the preceding we conclude that

$$f(\bar{\omega}) - f(\omega^*) \leq G_2(G_1 + \|\omega^*\|) + \frac{1}{2T}G_2^2(1 + \ln T). \tag{3.6.26}$$

□

Remark 3.6.5. With Theorem 3.3.4 we have shown that in a random iteration T, the resulting expected error is bounded by $O\Big(\frac{1 + \ln T}{T}\Big)$. However, it is interesting to note that SG-TPSVM enjoys an error bound similar to the SG-TSVM [25].

CHAPTER IV

Numerical Experiment

In this chapter we will establish the performance of our SG-TPSVM model. The experiments have been preformed on Python3 on a macOS with an Intel i5 Processor 2.3 GHz with Memory 8 GB 2133 MHz LPDDR3. In order to evaluate the performance of classifiers, we have used a 10-fold cross validation technique for all experiments. The results of each experiment show the average CPU time and standard deviation in all tables and highlight the best one. From now on we denote $\tau_1 = \tau_3$, $\tau_2 = \tau_4$, $\epsilon_1 = \epsilon_3$, and $\epsilon_2 = \epsilon_4$. To derive the non-linear case, we use the Gaussian kernel $K(x, y) = \exp\{-\sigma\|x - y\|^2\}$.

4.7 Synthetic dataset

The purpose of our SG-TPSVM model is to be able to handle noise-sensitive classifiers. To illustrate the noise insensitivity performance consider Figure ?? and Figure ??, where we take a two dimensional synthetic dataset with equal number of samples from two Gaussian distributions: $x_i, i \in \{i : y_i = 1\} \sim \mathcal{N}(\mu_1, \sum_1)$ and $x_i, i \in \{i : y_i = -1\} \sim \mathcal{N}(\mu_2, \sum_2)$, where $\mu_1 = [1, -3]^\top$, $\mu_2 = [-1, 3]^\top$ and $\sum_1 = \sum_2 = \begin{bmatrix} 0.2 & 0 \\ 0 & 3 \end{bmatrix}$. We add noise to the dataset, with each noise sample drawn from the Gaussian distribution $\mathcal{N}(\mu_n, \sum_n)$ where $\mu_n = [0, 0]^\top$ and $\sum_n = \begin{bmatrix} 1 & -0.8 \\ -0.8 & 1 \end{bmatrix}$. From Figure ??, the bar chart demonstrates the percentages of accuracy to classify from different sectors including SG-TPSVM, SG-TSVM [25] and PEGASOS [21] during 0 (free-noise) to 30 noise points. Overall, the SG-TPSVM achieves the best results in all cases. This implies that the SG-TPSVM was the strongest candidate for the method of classifying the data with noise corrupted.

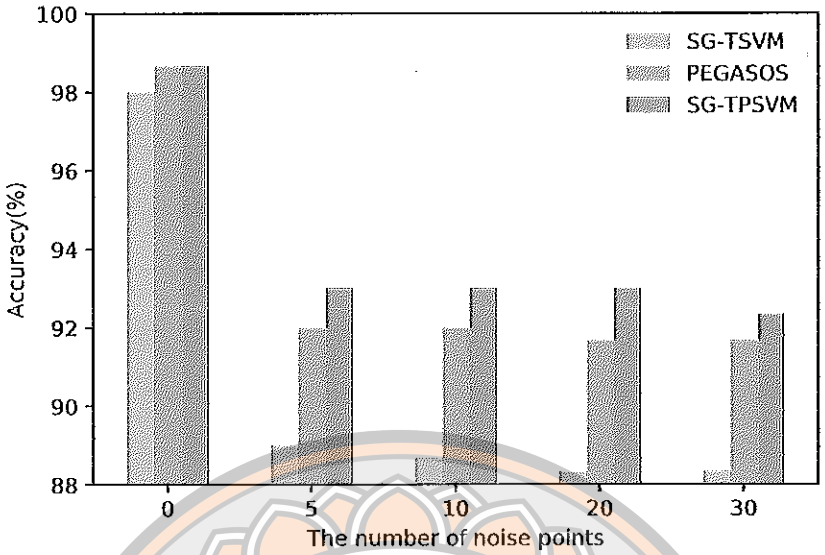
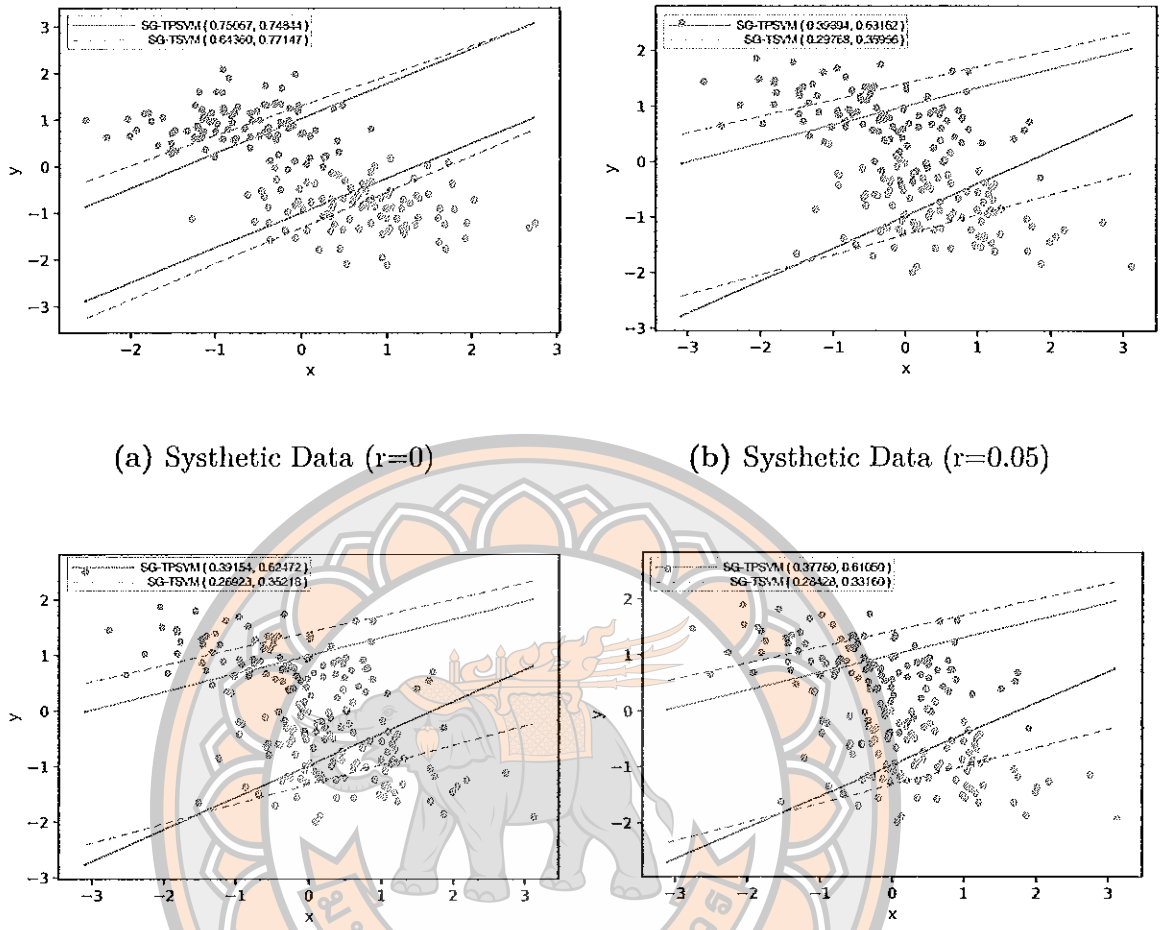


Figure 6: Bar graph of the accuracies for three algorithms on the 2-D artificial data.

Next result Figure ?? in this regard, showing when we increase the amount of noise from 0 to 20%, the hyperplanes of SG-TSVM diverge from 0.64360, 0.77147 to 0.28428, 0.33160 while the hyperplanes of our SG-TPSVM slightly changed. This means that the our SG-TPSVM model is insensitive to noise.



(a) Systhetic Data (r=0)

(b) Systhetic Data (r=0.05)

(c) Systhetic Data (r=0.1)

(d) Systhetic Data (r=0.2)

Figure 7: Illustration of the slopes of the separating hyperplanes.

4.8 UCI datasets

To test the performance of the proposed SG-TPSVM, experiments are carried out on eight UCI datasets from binary classification. All the data are summarized in Table 2. The features of each data set are corrupted by zero-mean Gaussian noise. For each feature, the ratio of the variance of noise denoted as r is set to be 0 (i.e., noise-free), 0.05, and 0.1.

In order to prove the efficacy of the proposed model, we have also compared the proposed formulation with SG-TSVM[25]. The comparison results are shown in Table 3 and Table 4. Here SG-TSVM(a), SG-TPSVM(a) and SG-TSVM(b), SG-TPSVM(b)

Table 2 The details of datasets.

datasets	No. of samples	Dimension
Australian	690	14
Banknote	1,372	5
Coil2000	900	86
Heart	270	13
Monk2	432	7
Pima Indians	768	8
Titanic	2,201	41
Twonorm	7,400	21

refer to linear and nonlinear cases, respectively. One can observe from Table 3 and Table 4 that the proposed model achieves better results in most cases. Figure 8 and Figure 9 show the average accuracy obtained from Table 3 and Table 4. It is clear that our SG-TPSVM has accuracy performance better than SG-TSVM. The computational time of both models are not different. The optimal parameters used in Table 3 and Table 4 are summarized in Table 5.

Table 3 Testing accuracy obtained from all discussed formulation in UCI datasets.

Datasets	r	Existing algorithms		Proposed algorithms	
		SG-TSVM(a)	SG-TSVM(b)	**SG-TPSVM(a)**	**SG-TPSVM(b)**
		Time (s)	Time (s)	Time (s)	Time (s)
Australian	0	86.81±3.27	76.09±7.73	**87.83±3.12**	84.49±3.37
		0.0639	0.2879	0.0830	0.4139
	0.05	85.65±3.52	76.67±6.53	**87.10±3.14**	84.78±3.68
		0.0627	0.2594	0.0862	0.4638
	0.1	86.23±2.69	74.78±8.58	**86.38±3.12**	84.78±3.26
		0.0602	0.2473	0.0831	0.4397
Banknote	0	83.75±2.27	97.15±2.27	84.62±2.90	**98.32±1.82**
		0.0656	0.4380	0.0432	0.7620
	0.05	88.12±2.24	96.86±2.77	84.40±2.98	**98.11±2.07**
		0.0610	0.4460	0.0417	0.4639
	0.1	88.34±2.66	96.50±2.23	83.38±3.20	**98.40±1.56**
		0.0599	0.4318	0.0402	0.4907
Coil2000	0	50.67±11.76	**93.29±0.78**	68.11±2.61	93.02±0.76
		0.0642	7.1999	0.0927	9.1320
	0.05	51.19±10.61	**94.03±0.71**	63.04±3.06	**94.03±0.71**
		0.0621	5.2486	0.0859	5.0169
	0.1	50.38±13.47	90.83±4.37	60.01±2.51	**94.03±0.17**
		0.0568	5.0152	0.0807	5.2470
Heart	0	83.33±5.30	83.70±6.46	80.00±5.79	**84.07±4.40**
		0.0627	0.3028	0.0844	0.3243
	0.05	82.59±4.98	81.48±7.03	82.96±4.44	**84.81±4.52**
		0.0665	0.3033	0.0863	0.3387
	0.1	83.33±5.80	82.59±5.98	83.70±4.12	**84.44±3.63**
		0.0612	0.2910	0.0870	0.3526

Table 4 Testing accuracy obtained from all discussed formulation in UCI datasets.

Datasets	r	Existing algorithms		Proposed algorithms	
		SG-TSVM(a)	SG-TSVM(b)	**SG-TPSVM(a)**	**SG-TPSVM(b)**
		Time (s)	Time (s)	Time (s)	Time (s)
Monk2	0	78.93±8.09	85.64±6.14	79.60±8.54	**89.58±4.91**
		0.0585	0.3128	0.0788	0.3434
	0.05	78.22±7.68	84.73±7.64	82.86±6.17	**87.04±6.83**
		0.0546	0.3359	0.0782	0.3666
	0.1	79.85±9.54	84.48±5.72	80.06±7.42	**88.42±7.71**
		0.0561	0.3527	0.0845	0.3783
Pima	0	72.40±4.75	**75.64±6.33**	73.55±4.80	74.35±5.38
		0.0644	0.3690	0.0800	0.3943
	0.05	72.27±6.11	73.95±4.95	**74.73±4.80**	73.56±5.95
		0.0549	0.3522	0.0775	0.4484
	0.1	72.65±4.61	73.42±6.42	**74.47±4.97**	73.16±5.87
		0.0527	0.3461	0.0740	0.3802
Titanic	0	74.47±3.47	66.96±2.04	76.56±3.20	**77.60±2.67**
		0.0397	0.2427	0.0791	0.5115
	0.05	75.15±2.50	68.01±2.62	**77.10±2.98**	76.92±2.71
		0.0590	0.3823	0.0789	0.5621
	0.1	75.15±3.07	67.92±2.69	76.60±2.45	**77.37±2.33**
		0.0560	0.4162	0.0833	0.5481
Twonorm	0	97.73±0.71	97.78±0.66	97.80±0.56	**97.81±0.63**
		0.0613	1.6700	0.0813	1.7512
	0.05	97.69±0.67	97.72±0.56	**97.85±0.63**	97.76±0.63
		0.0651	1.8604	0.0889	1.7798
	0.1	97.73±0.60	**97.78±0.55**	97.65±0.64	97.66±0.56
		0.0659	2.9027	0.0822	1.8999

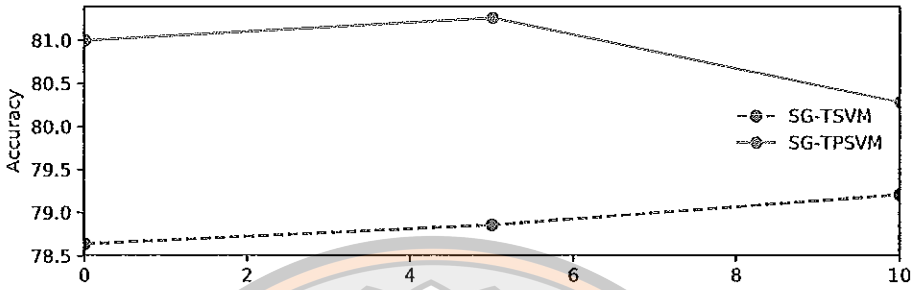


Figure 8: The average accuracy of SG-TSVM and SG-TPSVM (linear case) on UCI data sets.

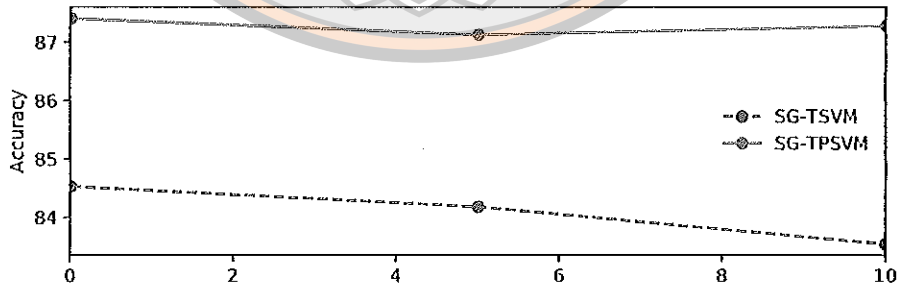


Figure 9: The average accuracy of SG-TSVM and SG-TPSVM (nonlinear case) on UCI data sets.

Table 5 The optimal parameters of SG-TSVM, SG-GTPPSVM.

Datasets	SG-TSVM(a)	SG-TSVM(b)	SG-TPSVM(a)	SG-TPSVM(b)
	c_1, c_2	c_1, c_2, γ	$c, \nu, \tau_1, \tau_2,$ ϵ_1, ϵ_2	$c, \nu, \tau_1, \tau_2,$ $\epsilon_1, \epsilon_2, \gamma$
Australian	1,1	0.1,1,0.01	1,1,2,0.01, 0.01,0.01	1,1,1,1, 1,1,0.1
Banknote	1,1	0.1,1,10	0.05,0.1,1,1, 1.5,0.5	1,1,1,1, 1,1,10
Coil2000	1,1	0.1,1,10	2,1.5,2,1, 1.5,0.5	1,1,1,1, 1,1,1
Heart	1,1	0.1,1,0.01	1,1,2,0.01, 0.01,0.01	1,1,1,1, 1,1,0.1
Monk2	1,1	0.1,1,1	0.01,1,2,0.01, 0.01,0.01	1,0.1,1,1, 1,1,1
Pima	1,1	0.1,1,0.1	0.01,1,2,0.01, 0.01,0.01	1,0.09,1,1, 1,1,0.1
Titanic	1,1	0.1,1,0.1	1,1,2,0.01, 0.01,0.01	1,1,1,1,1, 1,1,0.1
Twonorm	1,1	0.1,1,0.1	1,1,1,1.3, 0.01,0.01	1,1,1.3,1, 1,1,0.1

4.9 Large scale dataset

In this section, we show that our SG-TPSVM is capable to solve large scale problems. Note that this section has been solved on a machine equipped with an Intel CPU E5-2658 v3 at 2.20GHz and 256 GB RAM running Ubuntu Linux operating system. The usual SVMs are used with scikit-learn package [29]. The large scale datasets we used is presented in Table 6. For the nonlinear case, the reduced kernel [30] was used, and the kernel size was fixed to 100.

Table 6 The details of large scale datasets.

datasets	No. of samples	Dimension
Skin	245,057	3
SUSY	5,000,000	18
Kddcup	4,898,432	41
Hepmass	10,500,000	28

It is clear from Table 7 that the accuracy of SG-TPSVM is better than SVM and SG-TSVM. However, the experimental results on the two largest datasets, i.e. KDDCup and Hepmass, cannot be obtained due to the excessive memory requirement. This is because the implementation of SVM needs to store the entire training set in the main memory. Meanwhile SG-TSVM and SG-TPSVM only store a subset related to the iteration. Accordingly, SG-TSVM and SG-TPSVM is capable to solve largle scale problem.

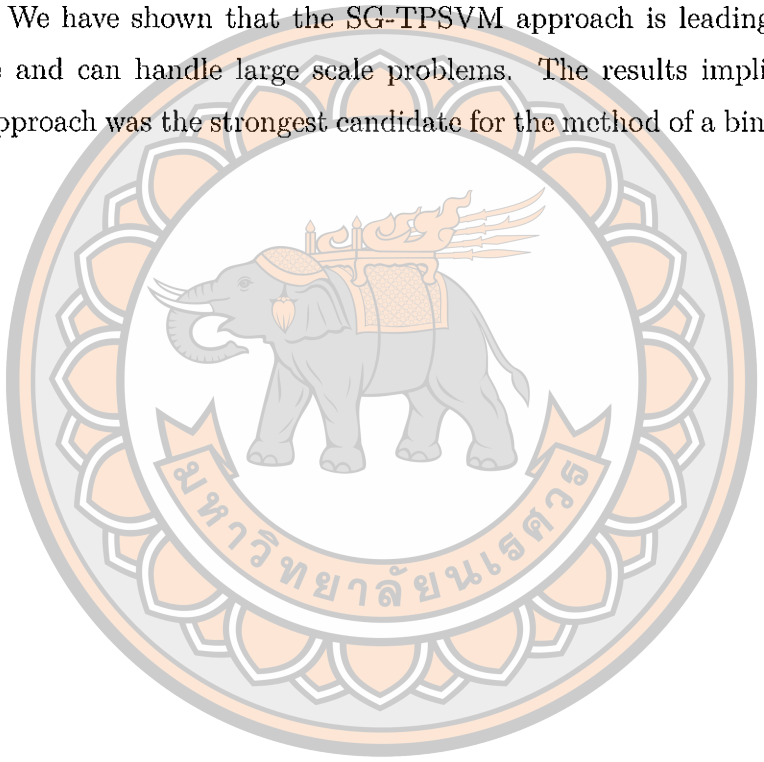
Table 7 The results for the large scale datasets.

	Existing algorithms			Proposed algorithms	
Datasets	SVM	SG-TSVM(a)	SG-TSVM(b)	**SG-TPSVM(a)**	**SG-TPSVM(b)**
Skin	78.87	85.23	84.70	92.64	**92.77**
Susy	**78.52**	75.09	68.61	75.91	68.70
Kddcup	*	95.24	93.19	96.72	**99.41**
Hepmass	*	81.10	79.59	**81.92**	79.16

CHAPTER V

CONCLUSION

In this thesis we have proposed a stochastic gradient descent method based on generalized pinball twin parametric support vector machine (SG-TPSVM). The efficiency of proposed method by using generated data and datasets imported from UCI is compared to SG-TSVM. The experimental results have shown that the accuracy performance of our method is better than the existing classifiers for different data scenarios. We have shown that the SG-TPSVM approach is leading to insensitivity from noise and can handle large scale problems. The results implies that the SG-TPSVM approach was the strongest candidate for the method of a binary classification problem.



REFERENCES

REFERENCES

1. Cortes C, Vapnik VN. Support vector networks. Machine Learning; 1995;20(3):273–97.
2. Cao L, Tay FEH. Financial forecasting using support vector machines. Neural Computional Applications. 2001;10(2):184–92.
3. Chathuramali KGM, Rodrigo R. Faster human activity recognition with SVM. ICT Emerg. Regions. 2012:197–03.
4. Byvatov E, Schneider G. Support vector machine applications in bioinformatics. Applications Bioinformat. 2003;2(2):67–87.
5. Huang Z, Chen H, Hsua CJ, Chen WH, Wu S. Credit rating analysis with support vector machines and neural networks: a market comparative study. Decision Support Systems. 2004;37:543–58.
6. Ince H, Trafalis TB. Support vector machine for regression and applications to financial forecasting. International Joint Conference on Neural Networks. IEEE-INNS-ENNS, Como, Italy. 2002.
7. Mukherjee S, Osuna E, Girosi F. Nonliner prediction of chaotic time series using a support vector machine. Proceedings of the 1997 IEEE Workshop on Neural Networks for Signal Processing, Amelia Island, FL, USA. 1997:511–20.
8. Jayadeva, Khemchandani R, Chandra S. Regularized least squares fuzzy support vector regression for financial time series forecasting. Expert Systems with Applications. 2009;36(1):132–38.
9. Mangasarian OL, Wild EW. Multisurface proximal support vector classification via generalized eigenvalues. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2006;28(1):69–84.
10. Jayadeva, Khemchandani R, Chandra S. Twin support vector machines for pattern classification. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2007;29(5):905–10.
11. Kumar MA, Gopal M. Least squares twin support vector machines for pattern classification. Expert Systems with Applications. 2009;36(4):7535–43.
12. Xu Y, Xi W, Guo R. An improved least squares twin support vector machine. Information Sciences. 2012;9(4):1063–71.
13. Peng X. TSVR: An efficient twin support vector machine for regression. Neural Networks. 2010;23(3):365–72.

14. Xu Y, Wang L. A weighted twin support vector regression. Knowledge Based Systems. 2012;33:92–1.

15. Hao PY. New support vector algorithms with parametric insensitive margin model. Neural Networks. 2010;23(1):60–3.

16. Peng X. A noval twin parametric-margin support vector machine for pattern recognition. Pattern recognition. 2011;44:2678–92.

17. Platt J. Fast training of support vector machines using sequential minimal optimization. In Advances in kernel methods-support vector learning, Cambridge, MA: MIT Press. 1999;185–8.

18. Joachims T. Making large-scale SVM learning practical. In Advances in Kernel Methods-Support Vector Learning. Cambridge. 1998;169–84.

19. Chang CC, Lin CJ. LIBSVM: A library for support vector machines [Internet]. Avaliable from: http://www.csie.ntu.edu.tw/ cjlin. 2001.

20. Fan RE, Chang KW, Hsieh CJ, Wang XR, Lin CJ. LIBLINEAR: a library for large linear classification. Journal of Machine Learning Research. 2008;9:1871–74.

21. Shwartz S, Singer Y, Srebro N, Cotter A. Pegasos:Primal estimated subgradient solver for SVM. Journal of Machine Learning Research. 2011;127(1):3–30.

22. Shao YH, Zhang C, Wang X, Deng N. Improvements on twin support vector machines. IEEE Transactions on Neural Networks. 2011;22(6):962–68.

23. Wang Z, Shao YH, Wu TR. A gabased model selection for smooth twin parametric margin support vector machin. Pattern Recognition. 2013;46(8):2267–77.

24. Tian YJ, Ping Y. Large-scale linear nonparallel support vector machine solver. Neural Networks. 2014;50:166–74.

25. Wang Z, Shao YH, Bai L, Li C, Liu L, Denge N. Insensitive stochastic gradient twin support vector machines for large scale problems. Information Sciences. 2018;462:114–31.

26. Huang X, Shi L, Suykens JA. Support vector machine classifier with pinball loss. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2014;36(5):984–97.

27. Reshma R, Khemchandani R, Pal A, Chandra S. Generalized pinball loss svms. Neurocomputing. 2018;36(5):322.

28. Shwartz S, David B. Understanding machine learning theory algorithms. Cambridge University Press. 2014

29. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M. Scikit-learn: Machine learning in Python. J. Machine Learning Research. 2011;12:2825–30.

30. Lee Y, Mangasarian O. RSVM: Reduced support vector machines. First SIAM International Conference on Data Mining. 2001:5–7