ROBUST SUPPORT VECTOR MACHINES IN CLASSIFICATION

BY

MOHAMMED M. AL-MEHDHAR

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

INDUSTRIAL AND SYSTEMS ENGINEERING

DECEMBER 2018

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN- 31261, SAUDI ARABIA
DEANSHIP OF GRADUATE STUDIES

This thesis, written by Mohammed Al-Mehdhar under the direction of his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN INDUSTRIAL AND SYSTEMS ENGINEERING.**

Mujahid Dec 24, 2018

Dr. Syed Mujahid
(Advisor)

24.12.2018

Dr. Shokri Z. Selim
(Member)

27-12-2018

Dr. Hesham K. Al-Fares
(Member)

Dr.Hesham K. Al-Fares
Department Chairman

Dr. Salam A. Zummo
Dean of Graduate Studies

8\1\19

Date

© Mohammed Al-Mehdhar

2018

To my parents Mehdhar and Fatimah, who are my sources of inspiration.

To my wife Manal, and my daughters Layan and Lamar, for all the love and support

ACKNOWLEDGMENTS

All praise to Allah for giving me the opportunity to pursue my master’s degree and for helping me throughout the last three years to successfully reach the end of this journey.

I would like to thank my thesis advisor Dr. Syed Mujahid for his ultimate support. He was always available to offer his advice and guidance throughout my thesis work. His passion for research has taught me a lot and motivated me to always go the extra mile. The experience I gained from him during the research and the process of writing the thesis is invaluable.

I also would like to thank the committee members: Dr. Shokri Selim and Dr. Hesham Al-Fares for their time and contribution. All the constructive comments and the encouragement they provided had a positive impact on my thesis research.

Finally, I express my gratitude to my parents who provided me with all the support I needed to finish my degree. I also pass my sincere gratefulness and appreciation to my wife for her endless support and patience throughout all the years of study and research. All this success would have been impossible without my family.

TABLE OF CONTENTS

ACKNOWLEDGMENTS......V

TABLE OF CONTENTS...... VI

LIST OF TABLES...... IX

LIST OF FIGURES......X

LIST OF ABBREVIATIONS...... XIII

ABSTRACT......XIV

ملخص الرسالة XVI

CHAPTER 1 INTRODUCTION......1

1.1 Data Mining & Analysis...... 2

1.2 Classification...... 3

1.3 Objective of The Thesis...... 5

1.4 Thesis Organization 7

CHAPTER 2 LITERATURE REVIEW 8

2.1 Support Vector Machine Models 8

2.2 Robust Support Vector Machine Models 16

2.3 Literature Gap 27

2.4 Problem Statement 33

CHAPTER 3 ROBUST MATHEMATICAL MODELS...... 36

3.1 Robust SVM Models for LDB...... 39

  3.1.1 Correntropic Support Vector Machine (C-SVM)...... 42

3.1.2 Correntropic Proximal SVM (C-PSVM) 43

3.1.3 Correntropic Membership Degree SVM (CMD-SVM) 43

3.2 Methods for NLDB 45

3.2.1 Dual Formulation 45

3.2.2 Feature Space and Kernel Trick 48

3.2.3 Generalized Representer Theorem 54

3.3 Robust SVM Models for NLDB 56

3.3.1 Correntropic Support Vector Machine (C-SVM) 56

3.3.2 Correntropic Proximal SVM (C-PSVM) 59

3.3.3 Correntropic Membership Degree SVM (CMD-SVM) 62

CHAPTER 4 SOLUTION METHODOLOGY 64

4.1 Model Convexity and Role of (σ) 65

4.2 Iterative Solution Method 72

4.3 Handling Parameter σ 73

4.4 Selection of Parameters (λ, γ, and σ) 83

4.4.1 Grid Search for γ and λ 83

4.4.2 Selecting σ 84

4.4.3 Grid Search Algorithm Using Iterative Solution Method 85

4.5 Solving Method for CMD-SVM Model 87

CHAPTER 5 NUMERICAL EXPERIMENTS 89

5.1 Cross Validation (CV) 90

5.2 Experiments Setup 94

5.3 Synthetic Illustration Problems 96

5.3.1 Segmented Blocks Experiment for LDB 97

5.3.2 Double Banana Experiment for NLDB 107

5.4 Real-Life Problems 116

CHAPTER 6 DISCUSSION AND CONCLUSION 125

6.1 Discussion and Highlights 125

6.1.1 Robustness in Classification 125

6.1.2 The Special Effect of σ 126

6.1.3 The Trade-off Between Time and Robustness 126

6.2 Conclusion 127

6.3 Future Research 128

6.3.1 GRT versus The Dual Formulation 128

6.3.2 A Closed Form for Determining $\sigma s, \sigma e$, and τ 128

REFERENCES 130

VITAE 133

LIST OF TABLES

Table 1: Typical data representation before processing 3

Table 2: Typical confusion matrix 92

Table 3: Details of the real-life data sets 117

Table 4: The prediction accuracies obtained by the 4 models for the 3 scenarios, using the real-life data sets. 119

Table 5: The total solution time in seconds for one complete repetition........................122

Table 6: The time (average time for C-SVM and C-PSVM) taken to solve the NLP problem for once..123

LIST OF FIGURES

Figure 1: A flow chart illustrating the major steps in a binary classification problem 6

Figure 2: Two different linear decision boundaries with their associated margins 9

Figure 3: The basic components to derive the SVM model 12

Figure 4: Incorporating the slack variables to derive the soft-margin SVM 15

Figure 5: Plot of the functions Hl(z) (___) , Hs(z) (- - - -), and Ts(z) (. . . .) with Ts = Hl – Hs [14] 20

Figure 6: The loss function (16) at different values of the scaling parameter σ 21

Figure 7: Plot of the rescaled hinge loss function at different scaling parameter η [16] .. 23

Figure 8: The Ramp loss function (black-solid) and smooth Ramp loss function (red-dashed) [18] 26

Figure 9: An illustration of two outliers imposing penalties 32

Figure 10: SVM best separating line before and after noise 34

Figure 11: PSVM best separating line before and after noise 35

Figure 12: The correntropic function in terms of the absolute error 38

Figure 13: The CLF at different values of σ 41

Figure 14: Data points for two classes are created in 2D (m=2) 49

Figure 15: Adding one space dimension (the vertical z-axis) and projecting the blue class along that new axis 50

Figure 16: The mapping from 2D to 3D allowed the two classes to be linearly separable 51

Figure 17: A plot of the C-SVM compact objective function using synthetic data. Critical areas of the graph are also identified. 66

Figure 18: Contours of the objective function ($w1 \times w2$) at different values of σ and for $\lambda = 10$ and the bias term $b = 1$. 68

Figure 19: Contours of the objective function (w_1×w_2) for σ=0.001 and λ=10 and the bias term b=1.. Multiple minima can be seen from the plot. 69

Figure 20: Contours of the objective function ($w1 \times w2$) for different values of λ and for given $\sigma = 0.001$ and the bias term $b = 1$. 70

Figure 21: Contours of the objective function (w_1×w_2) for λ=0.556 and for given σ=0.001 and bias term b=1. Multiple minima can be seen from the plot. 71

Figure 22: Two dimensional synthetic data points for two classes. Noise is represented by the blue points at the extreme top left corner and the red points at the extreme bottom right corner. 75

Figure 23: The predicted LDB obtained by a robust SVM model 76

Figure 24: The C-SVM model at sufficiently high values of σ produces the same LDB .78

Figure 25: The LDB produced by two different solution methodologies for the same parameters. This the onset of changing from pseudo-convex function to non-convex..79

Figure 26: While reducing σ, the LDB obtained by the proposed method is optimal and stable, whereas the LDB obtained by the existing method is bouncing and never reached the optimal orientation..80

Figure 27: The LDB obtained by both methods and very low values of σ 82

Figure 28: The structure of the data set used in the Segmented Blocks experiment.........98

Figure 29: The LDB obtained by the SVM model for 4 different scenarios 101

Figure 30: The LDB obtained by the PSVM model for 4 different scenarios 102

Figure 31: The LDBs obtained by the C-SVM model for 4 different scenarios.............103

Figure 32: The LDBs obtained by the C-PSVM model for 4 different scenarios...........104

Figure 33: The LDBs obtained by the CMD-SVM model for 4 different scenarios.......105

Figure 34: The NLDB obtained by the SVM for the 3 scenarios 109

Figure 35: The NLDB obtained by the PSVM for the 3 scenarios 110

Figure 36: The NLDB obtained by the C-SVM for the 3 scenarios...............................111

Figure 37: The NLDB obtained by the C-PSVM for the 3 scenarios 112

Figure 38: The NLDB obtained by the CMD-SVM for the 3 scenarios 113

Figure 39: An example of a continuous NLDB 114

Figure 40: An example of a discontinuous NLDB and the relevant classes 115

Figure 41: Line chart of the prection accuracies across the noise levels and per data set 120

LIST OF ABBREVIATIONS

SVM	:	Support Vector Machine
PSVM	:	Proximal Support Vector Machine
C-SVM	:	Correntropic Support Vector Machine
C-PSVM	:	Correntropic Proximal Support Vector Machine
CMD-SVM	:	Correntropic Membership Degree Support Vector Machine
LDB	:	Linear Decision Boundary
NLDB	:	Nonlinear Decision Boundary
CLF	:	Correntropic Loss Function
GRT	:	Generalized Representer Theorem
CV	:	Cross Validation
NLP	:	Non-Linear Programming

ABSTRACT

Full Name : Mohammed Mehdhar Al-Mehdhar
Thesis Title : Robust Support Vector Machines in Classification
Major Field : Industrial and Systems Engineering
Date of Degree : December 2018

Support Vector Machines (SVMs) are widely used in data classification. They are typically preferred due to their low generalized error. However, the conventional SVM models are sensitive to noise (outliers). Therefore, there is a need to develop robust data classification models that are insensitive to noise.

In this thesis, a total of 3 robust data classification models for binary classification are proposed. The three proposed models utilize the robust properties of the Correntropic Loss Function (CLF). Two of these models use the CLF as an error measure (C-SVM and C-PSVM), and the third uses the CLF as a membership degree (CMD-SVM). The proposed robust models are developed for the case of Linear Decision Boundary (LDB). For the case of Nonlinear Decision Boundary (NLDB), the proposed robust models were developed using the Generalized Representer Theorem (GRT).

An iterative solution methodology is developed to efficiently solve the proposed robust models. The proposed methodology exploits a very critical tuning parameter (σ) in the CLF, which controls the shape of the CLF. As a result, the objective function’s convexity is controlled throughout the solution process so that a local minimum is eventually guaranteed.

Finally, numerical experiments are conducted to illustrate the performance of the proposed robust models against the conventional SVM models. The experiments reveal that the proposed models are insensitive to noise and achieve higher prediction accuracies. However, this improved accuracy is achieved at the cost of the solution time. The introduction of the parameter σ into the robust models increases the total solution time due to the additional line search for σ. Nevertheless, to sum, based on the numerical experiments, the proposed methods works well in the presence and absence of noise.

ملخص الرسالة

الاسم الكامل: محمد محضار المحضار

عنوان الرسالة: نماذج آلة متجه الدعم المتينة في عمليات التصنيف

التخصص: هندسة النظم الصناعية

تاريخ الدرجة العلمية: أكتوبر 2018

أصبحت نماذج التصنيف المتينة (Robust Classification Models) في تطبيقات اليوم الصعبة والمتطورة ذو طلب عال. يتم استخدام نماذج آلة متجه الدعم (Support Vector Machines) (أ.م.د) على نطاق واسع في التصنيف وهي تفضل عادةً بسبب قوة نظريتها واشتقاقها الرياضي. ومع ذلك، فإن نماذج أ.م.د التقليدية حساسة للضوضاء "البيانات الشاذة" ولذلك، هناك حاجة إلى تطوير نماذج تصنيف قوية غير حساسة لضوضاء البيانات.

في هذه الرسالة، يتم اقتراح ثلاثة نماذج أ.م.د متينة للتصنيف الثنائي. تستخدم هذه النماذج الثلاثة دالة تستند إلى المفارقة وتسمى بدالة الفقدان الجزئي (Correntropic Loss Function) (د.ف.ج). اثنان من هذه النماذج يستخدم الدالة كدالة خسارة واضحة و هما الـ C-SVM و C-PSVM، أما الثالث فيستخدم الدالة كمقياس لدرجة العضوية و هو الـ CMD-SVM. تم تطوير النماذج المتينة المقترحة لحالة حدود القرار الخطية (Linear Decision Boundary). بالنسبة لحالة حدود القرار غير الخطية (Nonlinear Decision Boundary) ، تم تطوير النماذج المقترحة باستخدام نظرية التمثيل المعممة (Generalized Representer Theorem).

تم تطوير منهج حل تكراري مخصص لحل النماذج المتينة المقترحة بكفاءة عالية. تستغل منهجية الحل التكرارية المقترحة معلمة بالغة الأهمية في الـ د.ف.ج وهي معلمة ضبط تتحكم في شكل الدالة. و نتيجة لذلك، يتم التحكم في تحدب دالة الهدف طوال عملية الحل، و بالتالي يتم ضمان نقطة حرجة محلية في النهاية.

وأخيراً، تم إجراء تجارب رقمية لتقييم أداء النماذج المتينة المقترحة ضد نماذج أ.م.د التقليدية. تكشف التجارب أن النماذج المقترحة غير حساسة للبيانات الضوضائية كما أنها تحقق دقة تنبؤيه أعلى. ومع ذلك، يتم تحقيق هذه الدقة المحسنة على حساب الوقت المستغرق في الحل. يؤدي استخدام المعلمة σ في النماذج المتينة إلى زيادة الوقت

المستغرق في حل المسألة الكلية بسبب إضافة دورة بحثية جديدة حول المعلمة σ. وعلى الرغم من ذلك، فإن الخلاصة تكمن في أن النماذج المتينة المقترحة تعمل بشكل جيد في ظل وجود البيانات الضوضائية أو عدمها.

CHAPTER 1

INTRODUCTION

People rely on their experience and conjecture to predict the future before they ever used science to help them get better insights. That experience and intuition is related to something those people have learned or experienced before, which forms the content of what so called “history”. History can be seen as a huge database full of different types of data. This data can be carefully observed and analyzed to discover patterns and knowledge which can help us draw pictures of future events. The real challenge in the past was collecting, maintaining, and analyzing that extensive volume of data with limited tools and resources.

For the last two decades, the rapid technological evolution made data capturing, processing, and storage possible in huge volumes and incredible speed. This opened the gate to develop solutions that helped people and systems learn from existing data to make better decisions in the future. Since then, the concept of extracting knowledge from historical data has gained a huge interest from many researchers, entrepreneurs, and end-users around the globe. The demand of systems like image processing, voice recognition, and motion sensing are in steep growth, especially with the expanding arena of their applications. Corporates in many industries such as in IT security, digitalization, multimedia, and robotics are budgeting significant amounts of money for developing and improving those systems.

Learning from data originally came from two main fields known as Pattern Recognition and Machine Learning. The former originated from Engineering and the latter from computer science [1]. As the two fields grew, more fields were developed based on the background, types of applications, and the techniques used. One of the well-known fields is Data Mining, which is heavily linked to numerous applications in science, engineering, and business.

1.1 Data Mining & Analysis

Data mining is defined in [2] as a systematic process of analyzing given data from a system to discover patterns using algorithms in order to produce data analysis applications. The terms “Data Mining” is usually linked to the so called “knowledge discovery process”. This process uses discovery algorithms to recognize patterns and establishes models from the existing data [3]. The application areas for data mining are limitless and it currently takes important roles in sectors like banking, insurance, education, telecommunication, health, medicine, public, construction, engineering, and science [4].

The types of problems in data mining can be divided into two main categories. The first category is supervised learning where sufficient information about the data is given and future prediction is carried out using that given knowledge. The second category is unsupervised learning where no knowledge about the data to be worked on is available aside from the data points themselves. There are three major areas in data mining: clustering, classification, and regression. Clustering is considered to be a type of unsupervised learning, and classification & regression are known as supervised learning problems.

Before performing any data mining or analysis, the data needs to be organized in a format that is readable and easily understood. Table (1) represents an example of a hypothetical data set. The rows are the data instances whereas the columns represent the data features. This means the number of rows can be interpreted as the number of data points (samples) and the number of columns as the space dimension of the points. The last column (not a feature) is usually kept to identify the instance label in the case of supervised learning.

Table 1: Typical data representation before processing

Instance ID	**Feature 1 (*concentration*)**	**Feature 2 (*temperature*)**	⋯	**Feature *m* (*pressure*)**	**Label**
1	C1	T1	⋯	P1	Good
2	C2	T2	⋯	P2	Good
⋮	⋮	⋮	⋯	⋮	⋮
n	Cn	Tn	⋯	Pn	Bad

1.2 Classification

Classification is the process of learning from the available data to extract a pattern that will enable the user to classify a new instance. Classification is divided into three types: unary classification, binary classification, and multi-class classification. In unary classification, there is only one class and a new instance will be classified as “yes” (belongs to the class) or “no” (does not belong to the class). An example is the novelty detection problem, which classifies whether a machine’s status is “normal” or not. In this case, the data points used to build the classification model are the ones which indicate a “normal” machine status. This is because in nature, a machine’s status is “normal” most of the time hence, a sufficient number of data points can be obtained.

Binary classification classifies a data point into either one of two classes. One example of this type are classifying whether a person is well or sick. Another example is classifying whether a client will subscribe to a certain service or not. Binary and unary classifications are often confused for each other. The main difference between unary and binary classifications is the availability of the data. In novelty detection for example, it is difficult to collect data for the “abnormal” status because a machine fails once or twice a year. This makes it impractical to establish a second class with sufficient number of data points.

Multiclass classification is similar to binary classification but with more than two classes. An example is to classify whether a photo belongs to person X, person Y, or person Z. In some classification techniques, this type of classification is treated a set of multiple binary classification problems which eventually form a multiclass classification problem.

This thesis focuses on binary classification. The process of binary classification is summarized in Figure (1). It starts with collecting data from history and present it in a suitable format. Then a classification technique is chosen to train the classification model. Classification techniques are the algorithms that are applied on the available data to construct the classification models. This process is well known as *model training*. Several techniques are available to train the classification model such as artificial neural network, support vector machines, decision trees, and Bayesian Networks. Each technique has its own advantages and disadvantages w.r.t common performance measures like accuracy, speed of learning, tolerance to irrelevant attributes, and overfitting [5]. After training the model, the decision rule will be constructed. That decision rule is a function which takes a new data point as an input and assign it to one of the two classes.

1.3 Objective of The Thesis

The objective of this thesis is to develop binary classification models that are insensitive to data noise. The models will be developed upon the SVMs' framework. SVMs have unique characteristics (maximum margin) that distinguish the technique from other classification techniques. It is developed from a sound theory, and formulated as a convex problem. It also has low sensitivity to sample size and dimensionality. Moreover, SVMs have better generalization accuracy compared to the other classification techniques [5]. In addition to that, the proposed models will exploit the robustness of the CLF. It has been proven that the function is locally pesudoconvex. This key property will be used in developing the solution method for solving the proposed models. Thus, the proposed models will inherit the properties of SVM and CLF.

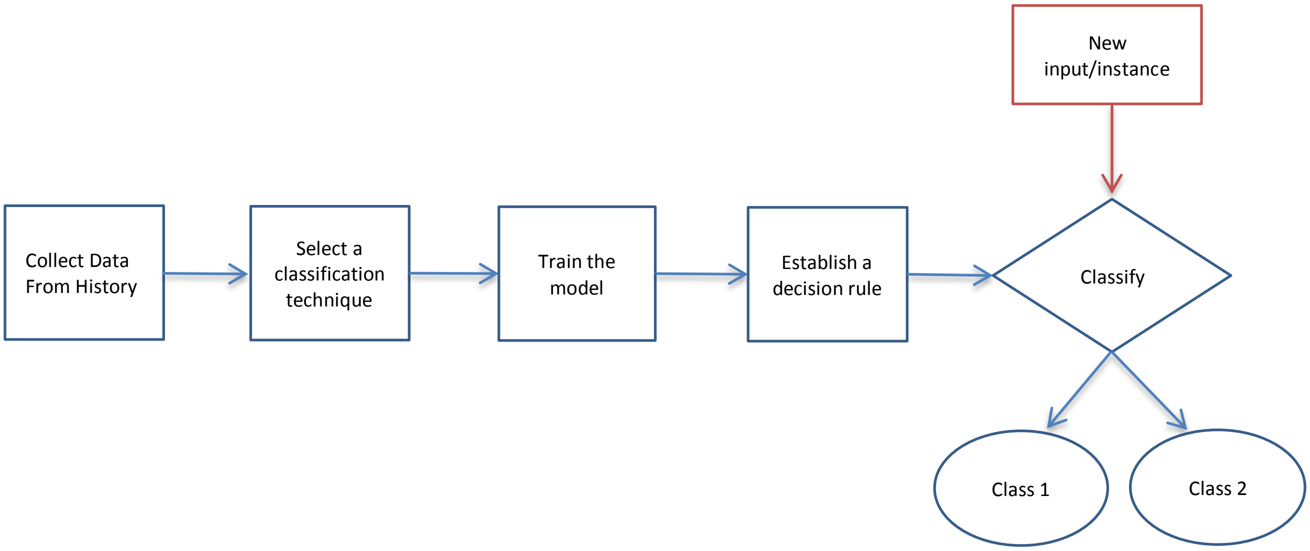

New input/instance
Collect Data From History
Select a classification technique
Train the model
Establish a decision rule
Classify
Class 1
Class 2

Figure 1: A flow chart illustrating the major steps in a binary classification problem

1.4 Thesis Organization

This rest of this thesis is organized as follows. In Chapter 2, the mathematical model of support vector machines is derived followed by a discussion on the robust SVM models from the literature. A literature gap is then presented in addition to the problem statement, which highlights the importance of this work. In Chapter 3, three robust SVM models are proposed and discussed followed by the solution methodologies in Chapter 4. After that, numerical experiments on well-known problems are presented in Chapter 5 to demonstrate the performance of the proposed models. Lastly, Chapter 6, a discussion on the critical concepts related to the thesis is presented, followed by conclusion and future research opportunities.

CHAPTER 2

LITERATURE REVIEW

In Section 2.1, different conventional formulations of the SVM models from the literature are presented. These conventional SVM models are not robust and they are sensitive to noise. Some improvements on the conventional models are presented in Section 2.2. These improvements will induce robustness and reduce noise sensitivity. Finally, a literature gap is presented to highlight the shortcomings in the literature and identify potential areas of improvement.

2.1 Support Vector Machine Models

Support Vector Machines are parametric methods, which are derived from the concept of maximum margin classification. The margin is defined as the Euclidean distance between the separating hyperplane (also known as the decision boundary) and the nearest data points [1][3]. The SVM defines the decision boundary, which separates the two classes into two distinct half-spaces, such that the margin is as large as possible. Such decision boundary will ideally result in the best classification accuracy. Figure (2) illustrates two possible decision boundaries and the associated margins. The decision boundary B has the largest margin, and it is the boundary that would be produced by the SVM model.

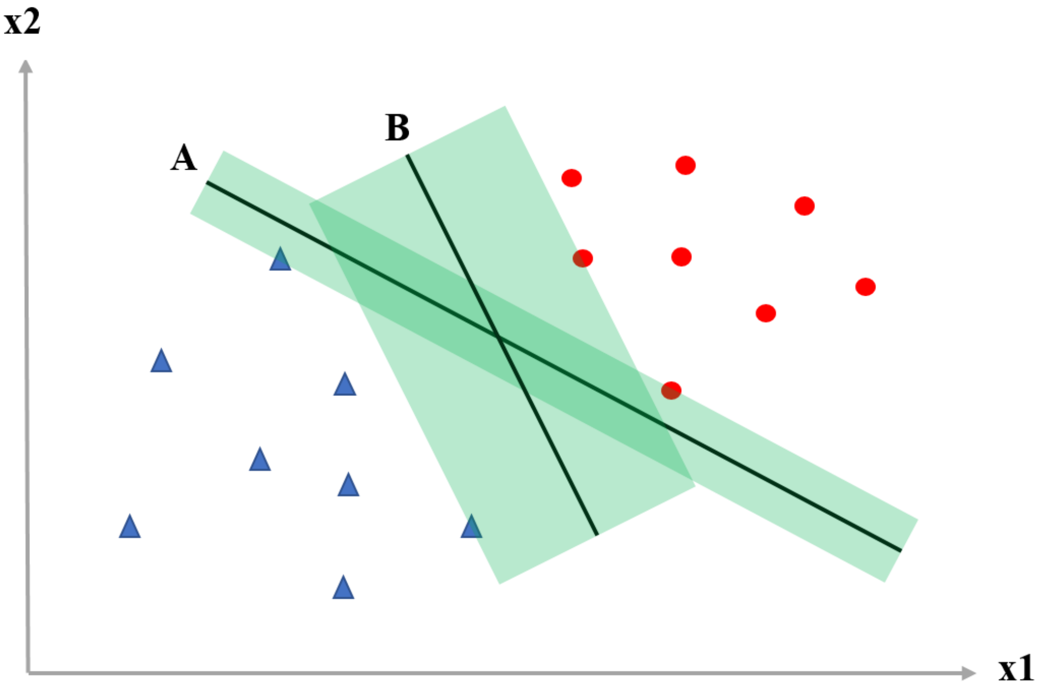


Figure 2: Two different linear decision boundaries with their associated margins

Data points are presented in pairs of $(\boldsymbol{x}_i, y_i)$, where $\boldsymbol{x}_i$ represents the data point in m dimensions, y_i is the class label defined as $y_i = -1 \; \forall \; i$'s which belong to class 1 and $y_i = +1 \; \forall \; i$'s which belong to class 2. The decision boundary is mathematically represented by the linear equation of the separating line (in higher dimensional space it is a hyperplane), which takes the form:

$$\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b = 0 \tag{1}$$

where $\boldsymbol{w} \in \mathbb{R}^{\boldsymbol{m}}$ and $b \in \mathbb{R}$ are parameters to be determined by the SVM model, and $\boldsymbol{x}$ is a data point. After obtaining the parameters ($\boldsymbol{w}$ and b) by solving the SVM model, the decision boundary can be used for classifying a new point into either of the two classes. This classification is carried out by the "rule function", to decide whether a point $\boldsymbol{x}$ belongs to Class 1 or Class 2. The rule function is defined as:

$$sign(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b) \tag{2}$$

where if the rule function is positive, then point $\boldsymbol{x}$ belongs to the positive class (Class 2), or if the rule function is negative, then $\boldsymbol{x}$ belongs to the negative class (Class 1).

Since the SVM aims to maximize the margin, the mathematical model can be constructed as an optimization problem that maximizes the margin between two disjoint classes. Figure (3) shows two classes separated by a line (decision boundary). The two dashed lines are known as "support hyperplanes" and they define the margin of the decision boundary. From the concept of SVMs as maximum margin classifiers, it is simple to realize that each support hyperplane must pass through at least one point. Those points which lie on these support hyperplanes are called "support vectors". The vector normal to the decision boundary is denoted as $\boldsymbol{w}$, which can be associated with the margin using a multiplier $\boldsymbol{r} \in \mathbb{R}^{+}$. It can be clearly seen that maximizing the margin is equivalent to maximizing $\boldsymbol{r}$. Taking a support vector from one class, say $\boldsymbol{x}^{-}$ from Class 1, and let $\boldsymbol{x}^{+}$ be the image of $\boldsymbol{x}^{-}$ on the other supporting hyperplane, along the direction w, defined as:

$$\boldsymbol{x}^{+} = \boldsymbol{x}^{-} + r\boldsymbol{w} \tag{3}$$

The equations of the two supporting hyperplanes are:

$$\boldsymbol{w}^{T}\boldsymbol{x}^{+} + b = +1 \tag{4a}$$

$$\boldsymbol{w}^{T}\boldsymbol{x}^{-} + b = -1 \tag{4b}$$

Substituting (3) in (4a) we get:

$$\boldsymbol{w}^{T}(\boldsymbol{x}^{-} + r\boldsymbol{w}) + b = +1$$

$$\Rightarrow \quad r\|\boldsymbol{w}\|^{2} + \boldsymbol{w}^{T}\boldsymbol{x}^{-} + b = +1 \tag{5}$$

Substituting (4b) in (5) leads to:

$$r\|\boldsymbol{w}\|^{2} - 1 = +1$$

$$\Rightarrow \; r = \frac{2}{\|\boldsymbol{w}\|^{2}} \tag{6}$$

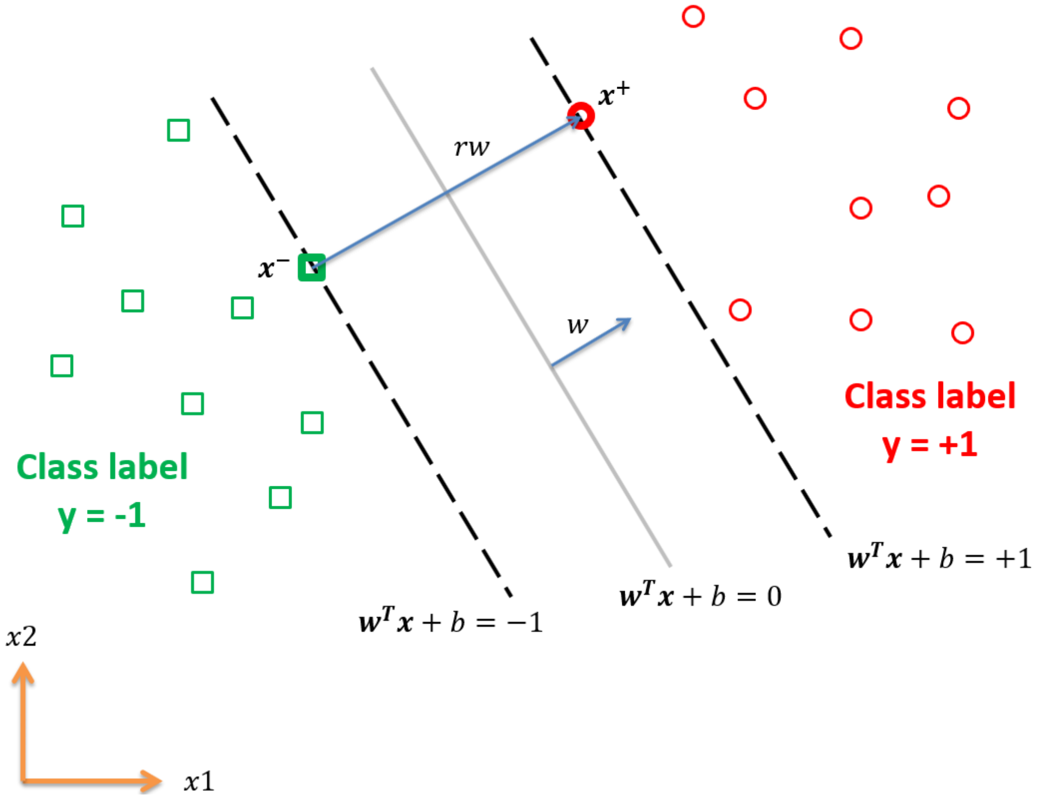

x^+
rw
x^-
w
Class label
y = +1
Class label
y = -1
$\mathbf{w}^T\mathbf{x} + b = +1$
$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b = -1$
$x2$
$x1$

Figure 3: The basic components to derive the SVM model

Now the SVM model can be formulated as follows:

$$\text{min.:} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 \tag{7a}$$

$$\text{s.t.:} \qquad \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}_{\boldsymbol{i}} + b \geq \ +1 \quad \forall\, i = 1,\ 2,\ \cdots, q\ \in Class\,1 \tag{7b}$$

$$\boldsymbol{w}^{\boldsymbol{T}}\ \boldsymbol{x}_{\boldsymbol{i}} + b \leq \ -1 \quad \forall\, i = q,\ q+1,\ \cdots, n\ \in Class\,2 \tag{7c}$$

where $\boldsymbol{w}$ and b are the unknowns, m is the number of features in the data, and n is the total number of data points. The two sets of constraints (7b) and (7c) can be combined into one set without changing the number of individual constraints as follows:

$$\text{min.:} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 \tag{8a}$$

$$\text{s.t.:} \qquad 1 - y_i(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}_{\boldsymbol{i}} + b) \leq 0 \quad \forall\, i = 1,\ 2,\ \cdots, n \tag{8b}$$

This model is feasible only when the data classes are linearly separable. In practical scenarios, linear separability is extremely difficult to find. To allow for a feasible solution to exist in such scenarios, a soft-margin SVM developed by Cortes and Vapnik [6] can be used. As shown on Figure (4), a slack variable for each data point is added to allow the data point to violate the constrains in (8b) and penalize that violation in the objective function. The soft -margin SVM is formulated as:

$$\text{min.:} \quad f = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n} s_i \tag{9a}$$

$$\text{s.t.:} \qquad 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \leq s_i \quad \forall\, i = 1, 2, \cdots, n \tag{9b}$$

$$s_i \geq 0 \quad \forall\, i \tag{9c}$$

where C is a parameter used as a scaling factor, and s_i represents the slack variable for data point $\boldsymbol{x}_i$. The second term in the objective function $(\sum_{i=1}^{n} s_i)$ is called “loss function”.

Another SVM model was proposed by Mangasarian and Wild in [7] that achieved a significant improvement on the solution time while maintaining a comparable solution accuracy. The model is referred to as "Proximal SVM (PSVM)" which utilizes two parallel support planes, where each plane crosses the center of mass of the associated data class. PSVM performs the classification of a new data point in a way such that the data point is closest to the associated plane of the class. On the other hand, the SVM classifies a data point with respect to the disjoint half-space around the decision boundary. The PSVM is modeled as follows:

$$\text{min.:} \quad \frac{1}{2}(\|\boldsymbol{w}\|^2 + b^2) + \frac{1}{2}C\|\boldsymbol{s}\|^2 \tag{10a}$$

$$\text{s.t.:} \quad 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) = s_i \quad \forall\, i = 1,2,\cdots,n \tag{10b}$$

where the notations are as previously defined for models (8) and (9).

It is important to mention that before the development of the PSVM, Suykens and Vandewalle [8] developed the Least Squares SVM (LS-SVM), formulated as:

$$\text{min.:} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{2}C\|\boldsymbol{s}\|^2 \tag{11a}$$

$$\text{s.t.:} \quad 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) = s_i \quad \forall\, i = 1,2,\cdots,n \tag{11b}$$

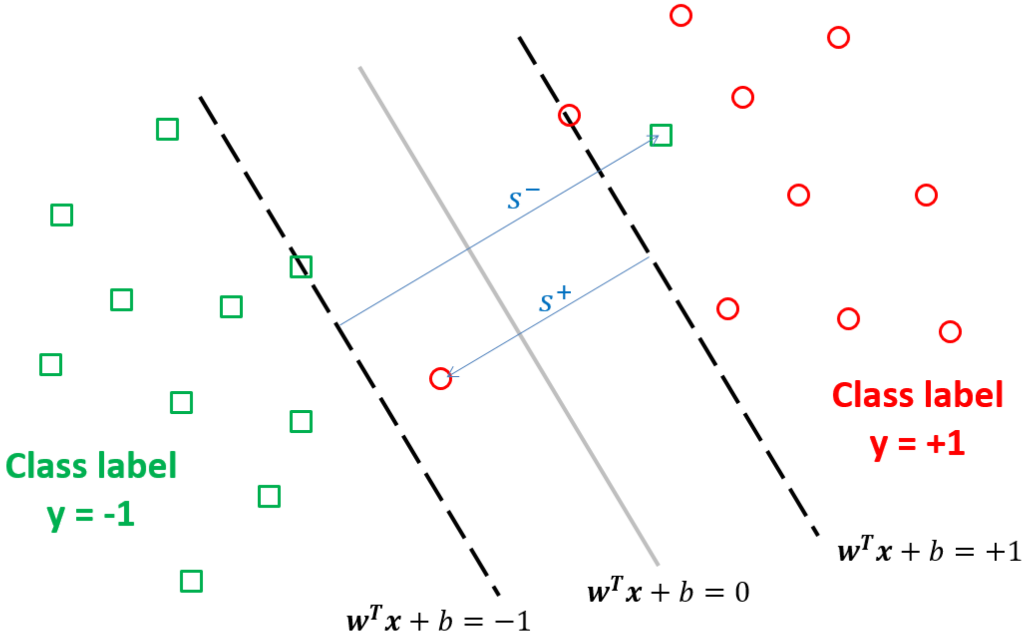

s^-
s^+
Class label
y = -1
Class label
y = +1
$\boldsymbol{w}^T\boldsymbol{x} + b = -1$
$\boldsymbol{w}^T\boldsymbol{x} + b = 0$
$\boldsymbol{w}^T\boldsymbol{x} + b = +1$

Figure 4: Incorporating the slack variables to derive the soft-margin SVM

which is the same as the PSVM but without the bias term "b^2" in the objective function. The objective function in LS-SVM is not strongly convex like in the PSVM [7] and that is because of the absence of the term "b^2" in the objective function. Furthermore, the speed of the PSVM solution algorithm is better than that of the LS-SVM, which is a differentiating factor for the PSVM [7].

The soft-margin SVM (it will call be called as SVM from now on for convenience) and PSVM models are attractive due to their convexity which makes the problems easy to solve. However, a major weakness of these models is their sensitivity to outliers. In soft-margin SVM and PSVM, the error is computed by the loss functions which are linear and quadratic, respectively. Those loss functions return high values for outliers, which influence the decision boundary and may result in poor classification accuracy.

2.2 Robust Support Vector Machine Models

One major reason for noise sensitivity in SVM is treating all data points with the same importance, without differentiating outliers from regular points. To overcome this obstacle, Fuzzy Support Vector Machine (FSVM) was proposed. FSVM allocates a weight to each individual data point to adjust its contribution in the model [9]. The objective function from the SVM model is reformulated as $\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\alpha_i s_i$, where α_i is called "fuzzy membership" and it determines the importance of data point $\boldsymbol{x}_i$. The same concept was applied by Meng et al. on the PSVM model to induce robustness [10]. The model is called as Weighted Support Vector Machine (WSVM) and the second term

$\frac{1}{2}\alpha\|\boldsymbol{s}\|^2$ becomes $\frac{1}{2}\|\boldsymbol{\alpha s}\|^2$, where, $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \cdots, \alpha_n]$. Each weight value α_i in the WSVM is a function of the data point distance to the class center, and it can be mathematically expressed as $\alpha = 1 - \frac{d}{R+\gamma}$, where the parameter d can be defined as the Euclidian distance from the associated class center, R is the radius of the class, and γ is a positive real number to ensure that $\alpha \neq 0$. Although this weighing method enhanced the PSVM robustness, such a simple formula to calculate the weights does not necessarily capture the real demography of the data. One improvement upon this shortcoming is using sophisticated statistical methods to calculate the membership weights. Principal Component Analysis (PCA) was utilized by Heo & Gader [11] to calculate the reconstruction error, which measures the relationship of a data point to the whole data structure. The idea is to perform PCA on each class to come up with the principal components. Then the reconstruction error is used to calculate the membership of the data points. The simple expression of the reconstruction error for a centered data point is expressed as: $E(x) = \|x - \boldsymbol{P}_X\boldsymbol{P}_X^T x\|^2$; and $\boldsymbol{P}_X$ is a matrix consisting of the principal components of the covariance matrix of X.

The FSVM can be further improved by utilizing the concept of bilateral weighing. The bilateral-weighted fuzzy SVM (BW-FSVM) introduced in [12] aims to reduce the effect of noise in classification problems. The method restructures the data sets from $G = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_n, y_n)\}$ to $G' = \{(\boldsymbol{x}_1, +1, \gamma_1), (\boldsymbol{x}_1, -1, 1 - \gamma_1), (\boldsymbol{x}_2, +1, \gamma_2), (\boldsymbol{x}_2, -1, 1 - \gamma_2), \cdots, (\boldsymbol{x}_n, +1, \gamma_n), (\boldsymbol{x}_n, -1, 1 - \gamma_n)\}$, where γ_i is called the membership degree (weight) and it constructs the association level of that data point to its assigned class. As a result, the BW-FSVM model is built as follows:

$$\text{min.:}\quad f = \frac{1}{2}\|\boldsymbol{w}\|^2 + \ C\sum_{i=1}^{n}(\ \gamma_i s_i + (1-\gamma_i)s_i'\ ) \tag{12a}$$

$$\text{s.t.:}\qquad 1-(\boldsymbol{w}^T\boldsymbol{x}_i + b) \le s_i \quad \forall\, i = 1,2,\cdots,n \tag{12b}$$

$$1+(\boldsymbol{w}^T\boldsymbol{x}_i + b) \le \ s_i' \quad \forall\, i = 1,2,\cdots,n \tag{12c}$$

$$s_i, s_i' \ge 0 \quad \forall\, i \tag{12d}$$

Despite the contribution by Wang et al. in [12], the BW-FSVM still performs on a simple weight assignment platform. To impose more robustness into the model, Yang et al. [13] incorporated a bilateral truncated hinge loss function abbreviated as BTL-RSVM and presented as:

$$\text{min.:}\quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \ C\sum_{i=1}^{n} T(\boldsymbol{w}, b, \boldsymbol{x}_i) \tag{13a}$$

$$T(\boldsymbol{w}, \boldsymbol{x}_i, b) = \min\{1, [1-(\boldsymbol{w}^T\boldsymbol{x}_i + b)]_+\} + \ \min\{1, [1+(\boldsymbol{w}^T\boldsymbol{x}_i + b)]_+\} \tag{13b}$$

This enhanced model was inspired by some earlier improvements on the well-known hinge loss function [14]. This function is a common function used in classification and regression (because of its simplicity and early discovery). The hinge loss function can be written as:

$$H_l(z) = (1-z)_+\,,$$

where $(a)_+ = \begin{cases} a & if\ a \ge 0 \\ 0 & otherwise \end{cases}$. To incorporate the hinge loss function into SVM, the model is written as:

$$\text{min.}: \; f = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n} H_l\big(y_i f(\boldsymbol{x}_i)\big) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\big(1 - y_i f(\boldsymbol{x}_i)\big)_+ \tag{14}$$

The hinge loss function can be truncated to become bounded and avoid extreme cost values caused by outliers [14]. As a result, the model robustness will be improved. The truncated hinge loss $T_s(z)$ developed by Wu and Liu [14] is expressed as

$$T_s(z) = H_l(z) - H_s(z),$$

where $H_s(z) = (1 - z)_+$.

$$\Rightarrow \; T_s(z) = (1 - z)_+ - (s - z)_+ \tag{15}$$

The truncated loss function shown in Figure (5) prevents the error values of $z < s$ to be inflated and assign high importance to noise.

Because of the simplicity and effectiveness of the hinge loss function, it can be deployed in a wide range of formulations to suppress noise sensitivity. For instance, the hinge loss function can be utilized as a similarity measure term built in another loss function. This is called the "hinge-loss-based loss function" [15] and it is formulated as:

$$l_\sigma(y,t) = \sigma^2\left(1 - e^{\frac{-H_l^2}{\sigma^2}}\right) = \sigma^2\left(1 - e^{\frac{-(1-yt)^2{}_+}{\sigma^2}}\right); y \in \{-1,+1\} \tag{16}$$

Figure (6) shows the loss function $l_\sigma(y,t)$ at different scaling parameter σ.

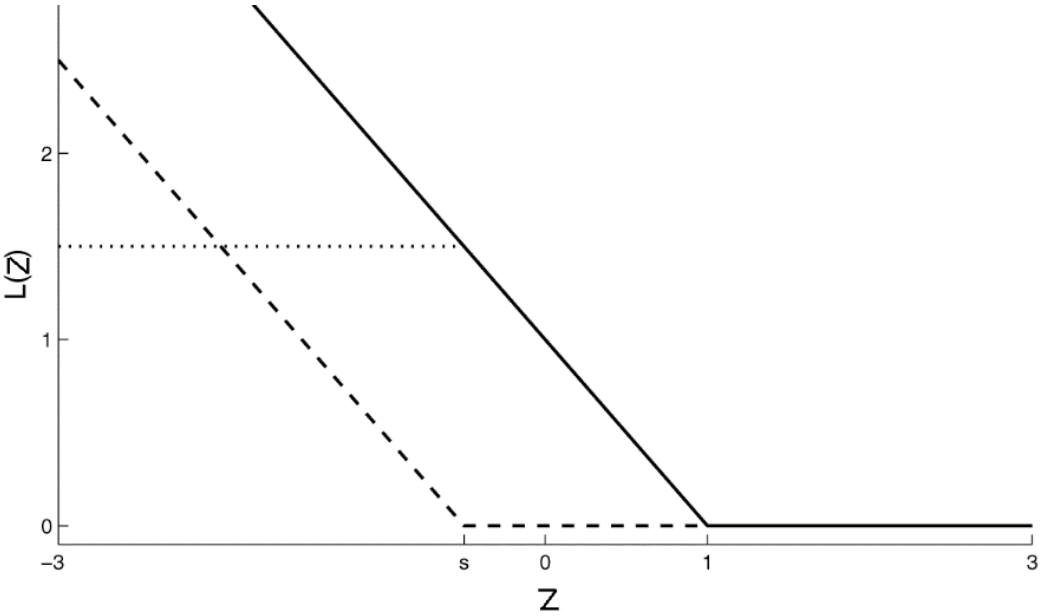


Figure 5: Plot of the functions Hl(z) (___) , Hs(z) (- - - -), and Ts(z) (. . . .) with Ts = Hl – Hs [14]

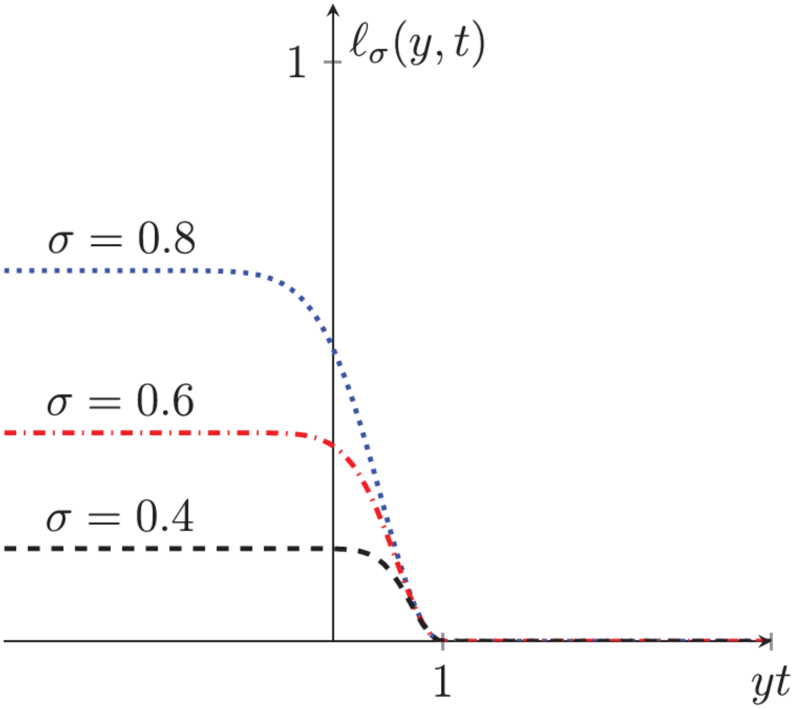


Figure 6: The loss function (16) at different values of the scaling parameter σ

The model can be formulated per the following:

$$\text{min.}: \ f = \alpha\|\boldsymbol{w}\|_2^2 + \frac{1}{n}\sum_{i=1}^{n} l(y_i\,, \boldsymbol{w}^T\boldsymbol{x}_i + b) \tag{17a}$$

Or

$$\text{min.}: \ f = \alpha\|\boldsymbol{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n}\sigma^2\left(1 - e^{\frac{-\left(1-y_i\left(w^T x_i+b\right)\right)_+^2}{\sigma^2}}\right) \tag{17b}$$

where α is a scaling factor that assigns the contribution of the relevant terms in the objective function. In a generally similar model structure, the hinge loss function was re-scaled and smoothed to allow for a more general form of the function as illustrated in Figure (7) [16]. The rescaled hinge loss function is formulated to be a function of the hinge loss function as $H_{re}(\mathrm{z}) = \beta\left(1 - e^{-\eta\, H_l(\mathrm{z})}\right)$; where $\beta = (1 - e^{-\eta})^{-1}$ is a normalizing constant and η is a smoothing parameter. The mathematical model in [16] is constructed as follows:

$$\text{min.}: \ f = \frac{1}{2}\|\boldsymbol{w}\|^2 + \ C\ \sum_{i=1}^{n} H_{re}(\mathrm{z}_i) = \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \ C\beta\ \sum_{i=1}^{n}\left[1 - e^{-\eta\,(1-y_i z_i)_+}\right] \tag{18}$$

The hinge loss function was also implemented in statistical models utilizing the concept of "Least Median Regression". The Hinge loss function $H_l(z)$ in model (14) is replaced with Class Conditional Median Loss function (CCML). The CCML is written such that:

$$CCML = \frac{1}{2}C\ \left[\ \underset{i\in class\ 1}{median}\left(1 - (1)f(\boldsymbol{x}_i)\right)_+ + \ \underset{i\in class\ 2}{median}\left(1 - (-1)f(\boldsymbol{x}_i)\right)_+\ \right] \tag{19}$$

in addition to a balancing constraint [17].

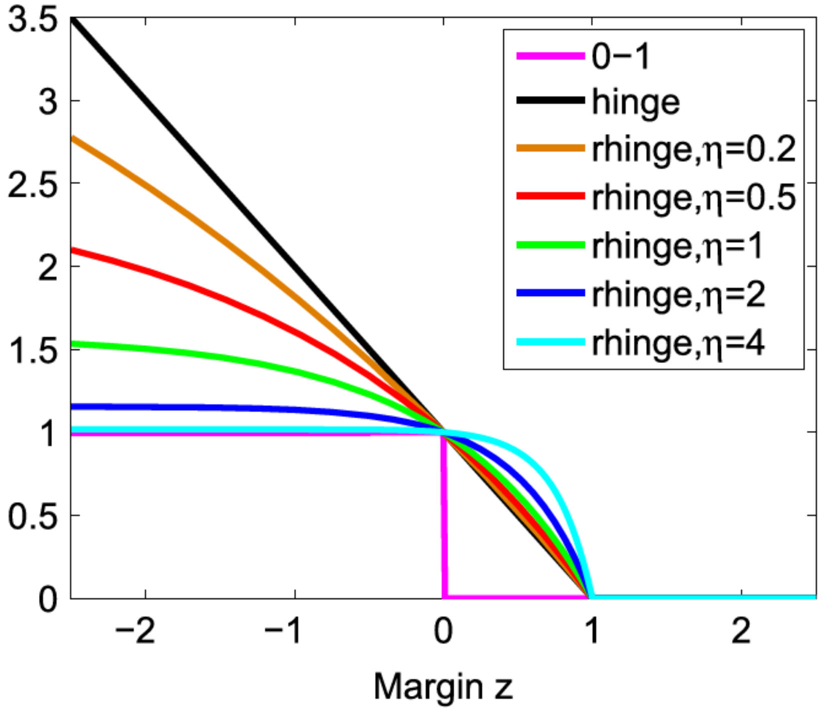


Figure 7: Plot of the rescaled hinge loss function at different scaling parameter η [16]

Although truncated loss functions provide good noise insensitivity, the optimization problem may be difficult to solve in general. This is mainly because these truncated loss functions are not differentiable. Wang et al. [18] developed a smooth ramp loss function which integrates the concave Huber loss and convex Huber loss functions. Figure (8) shows the smooth Ramp loss function which is bounded, continuous, and twice differentiable. The model is formulated as follows:

$$\text{min.}: \; f = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n} H_1^{hu}\left(y_i f(\boldsymbol{x}_i)\right) + C\sum_{i=1}^{n} H_0^{hu}\left(y_i f(\boldsymbol{x}_i)\right) \tag{20}$$

where $H_0^{hu}(z) = \begin{cases} 0 & if\ z > h, \\ -\frac{(h-z)^2}{4h} & if\ |z| \le h, \\ z & if\ z < -h, \end{cases}$, $H_1^{hu}(z) = \begin{cases} 0 & if\ z > 1+h, \\ \frac{(1+h-z)^2}{4h} & if\ |1-z| \le h, \\ 1-z & if\ z < 1-h, \end{cases}$

and h is Huber parameter. The proposed model in [18] achieved better generalization performance than the classical SVMs.

In addition to the above approaches, there are approaches that focus on the insensitivity w.r.t the spread of the data. For example, SVM takes into consideration the spread of the data within each class and reacts to changes in data spread along any direction [19]. To overcome this drawback, Jebara & Shivaswamy [19] developed the Relative Margin Machines (RMMs), which is a modification to the SVM, by adding the following constrain $\frac{1}{2}(\boldsymbol{w}^T\boldsymbol{x}_i + b)^2 \le \frac{B^2}{2} \quad \forall\, i$, where B is a constant with the range $B \ge 1$. RMM is insensitive to the data spread parallel to the decision boundary, which induces robustness to affine scaling. This indicates that the RMM model will position the decision boundary in a way that maximizes the margin only in the direction relative to the spread of the data. Moreover, RMM can be enhanced to deal with complicated data distributions.

This can be achieved by incorporating a loss function in the model such as the pinball loss function [20].

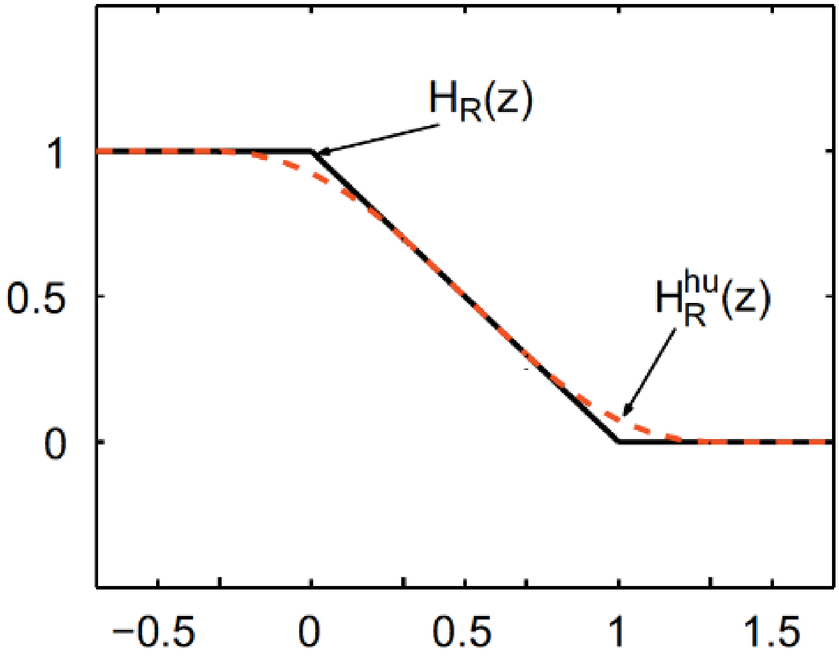


Figure 8: The Ramp loss function (black-solid) and smooth Ramp loss function (red-dashed) [18]

2.3 Literature Gap

Despite the improvements presented in the literature to reduce noise sensitivity in SVM’s, there are still some aspects which have not been sufficiently improved. The robust models that relied on the membership degrees are limited to basic calculations to assign the weights to the points. For some complicated data structures, those methods may not be sufficiently effective to reduce the role of outliers. Additionally, those models would require significant computational effort to calculate the weights in advance before solving the model. Therefore, using loss functions in the SVM models has been preferred and gained more interest from researchers.

The loss function in the objective function of the optimization model aims to penalize incorrectly classified points. The model becomes robust when outliers are not penalized (or only slightly penalized). This will prevent the outliers from playing a significant role in determining the decision boundary. One way to obtain this behavior in loss functions is truncation. Truncated loss functions limit their output to constant values beyond certain argument values in its domain, thus limiting the penalty in the objective function. However, the main drawback of truncated loss functions is smoothness. Truncated loss functions are normally not smooth, hence not continuously differentiable. That may cause the SVM optimization problem to be difficult and inefficient to solve.

To overcome the shortcoming of non-smoothness in some loss functions, they were modified to be smooth and continuously differentiable. Despite these improvements, the modified models are still non-convex and require global optimization algorithms to solve.

The robust SVM work by Feng, et al. [15] starts with a reasonable model formulation which seems to be promising for achieving their objective of designing a robust and smooth loss function. However, the mathematical derivations afterwards and the proposed problem solution have some gaps. The problem is initially formulated as:

$$\min_{\boldsymbol{u} \in \mathbb{R}^n, b \in \mathbb{R}} \quad \frac{1}{n}\sum_{i=1}^{n} \varphi((1 - y_i \boldsymbol{K}_i^T \boldsymbol{u} - y_i b)_+^2) \ + \lambda \boldsymbol{u}^T \boldsymbol{K} \boldsymbol{u}$$

$$= \min_{\boldsymbol{u} \in \mathbb{R}^n, b \in \mathbb{R}} \quad \frac{1}{n}\sum_{i=1}^{m} \sigma^2 \left(1 - e^{\frac{-\left(1 - y_i \boldsymbol{K}_i^T \boldsymbol{u} - y_i b\right)_+^2}{\sigma^2}} \right) + \lambda \boldsymbol{u}^T \boldsymbol{K} \boldsymbol{u} \tag{21}$$

where $\boldsymbol{u}$ is the Lagrangian variable, and $\boldsymbol{K_i}$ is the ith column of the kernel matrix. This formulation is for non-linear decision boundaries, which has not been tackled yet in this thesis but will be discussed in detail in later chapters. For the sake of simplicity, we rewrite (21) for a linear decision boundary (LDB) in terms of $\boldsymbol{w}$ and b and the problem will be changed to:

$$\min_{\boldsymbol{w} \in \mathbb{R}^m, b \in \mathbb{R}} \quad \frac{1}{n}\sum_{i=1}^{n} \sigma^2 \left(1 - e^{\frac{-\left(1 - y_i \boldsymbol{w}^T x_i - y_i b\right)_+^2}{\sigma^2}} \right) + \lambda \|\boldsymbol{w}\|^2 \tag{22}$$

Taking the partial derivatives for (22) with respect to $\boldsymbol{w}$ and b:

$$\frac{\partial R(\boldsymbol{w},b)}{\partial \boldsymbol{w}} = \frac{1}{n}\sum_{i=1}^{n} \left[2\left(1 - y_i(\boldsymbol{w}^T \boldsymbol{x_i} + b)\right)_+ e^{\frac{-\left(1 - y_i \boldsymbol{w}^T x_i - y_i b\right)_+^2}{\sigma^2}} (-y_i \boldsymbol{x_i}) \right] + 2\lambda \boldsymbol{w} = 0 \tag{23a}$$

$$\frac{\partial R(\boldsymbol{w},b)}{\partial b} = \frac{\sigma^2}{n}\sum_{i=1}^{n} \left[0 - \frac{(-1) \times 2 \times \left(1 - y_i(\boldsymbol{w}^T \boldsymbol{x_i} + b)\right)_+}{\sigma^2} (-1) y_i e^{\frac{-\left(1 - y_i(\boldsymbol{w}^T \boldsymbol{x_i} + b)\right)_+^2}{\sigma^2}} \right] + 0 = 0 \tag{23b}$$

Now let $A_i = 2e^{\frac{-\left(1-y_i\left(w^T x_{\boldsymbol{i}}+b\right)\right)_+^2}{\sigma^2}}$, then equations (21) and (22) can be reduced to:

$$\frac{\partial R(\boldsymbol{w}, b)}{\partial \boldsymbol{w}} = \frac{1}{n}\sum_{i=1}^{n}\left[A_i y_i \boldsymbol{x}_i\left(1 - y_i(\boldsymbol{w}^T \boldsymbol{x}_{\boldsymbol{i}} + b)\right)_+\right] - 2\lambda \boldsymbol{w} = 0 \qquad \text{(24a)}$$

$$\frac{\partial R(\boldsymbol{w}, b)}{\partial b} = \sum_{i=1}^{n}\left[y_i A_i\left(1 - y_i(\boldsymbol{w}^T \boldsymbol{x}_{\boldsymbol{i}} + b)\right)_+\right] = 0 \qquad \text{(24b)}$$

Equations (24a) and (24b) can be written in a kernelized form as:

$$\frac{\partial R(\boldsymbol{u}, b)}{\partial \boldsymbol{u}} = \frac{1}{n}\sum_{i=1}^{n}\left[A_i y_i \boldsymbol{K}_i\left(1 - y_i(\boldsymbol{K}_i^T \boldsymbol{u} + b)\right)_+\right] - \lambda \boldsymbol{K}\boldsymbol{u} = 0 \qquad \text{(25a)}$$

$$\frac{\partial R(\boldsymbol{u}, b)}{\partial b} = \sum_{i=1}^{n} y_i A_i\left(1 - y_i(\boldsymbol{K}_i^T \boldsymbol{u} + b)\right)_+ = 0 \qquad \text{(25b)}$$

Equation (25a) is found to be matching with that of Feng, et al. However, Equation (25b) was derived by the authors to be:

$$\sum_{i=1}^{n} A_i(y_i - \boldsymbol{K}_i^T \boldsymbol{u} - b) = 0 \qquad \text{(26)}$$

The two equations (25b) and (26) are not the same and they yield to different solutions. The authors further utilized equations (25a) and (25b) in [15] to develop the following iteratively reweighted algorithm to solve the model (Q1):

$$(\boldsymbol{u}^{r+1}, b^{r+1}) = \underset{\boldsymbol{u}\,\in\,\mathbb{R}^n,\, b\,\in\,\mathbb{R}}{argmin} \sum_{i=1}^{n} \omega_i^{r+1}(y_i - \boldsymbol{K}_i^T \boldsymbol{u} - b)_+^2 + \lambda \boldsymbol{u}^T \boldsymbol{K}\boldsymbol{u} \qquad \text{(27a)}$$

where

$$\omega_i^{r+1} = e^{\frac{-\left(y_i - \boldsymbol{K}_i^T \boldsymbol{u}^r - b^r\right)_+^2}{\sigma^2}}, \qquad i = 1, 2, \dots, n \tag{27b}$$

and r is the iteration number.

Again, it is easier to illustrate the performance of model (27) in the LDB form instead of the kernelized form. Therefore, model (27) is rewritten in terms of $\boldsymbol{w}$ and b as follows:

$$(\boldsymbol{w}^{r+1}, b^{r+1}) = \underset{\boldsymbol{w} \in \mathbb{R}^m, b \in \mathbb{R}}{argmin} \sum_{i=1}^{n} \omega_i^{r+1} (y_i - \boldsymbol{w}^T \boldsymbol{x_i} - b)_+^2 + \lambda \|\boldsymbol{w}\|^2 \tag{28a}$$

where

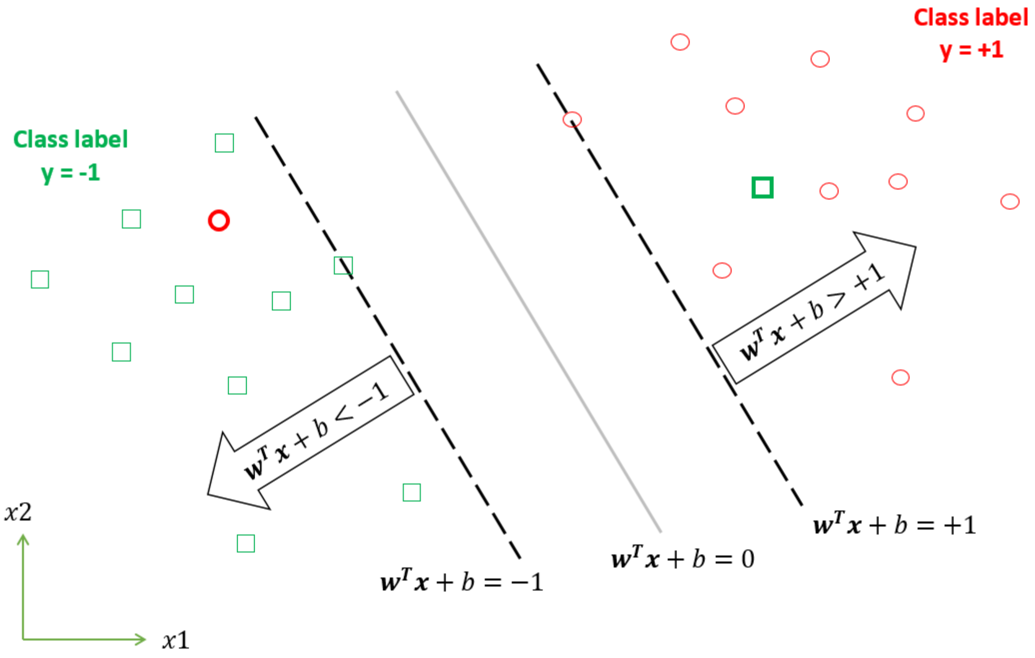
$$\omega_i^{r+1} = e^{\frac{-\left(y_i - \boldsymbol{w}^{r^T} \boldsymbol{x_i} - b^r\right)_+^2}{\sigma^2}}, \qquad i = 1, 2, \dots, n. \tag{28b}$$

and r is the iteration number.

The loss function $\sum_{i=1}^{n} \omega_i^{r+1} (y_i - \boldsymbol{w}^T \boldsymbol{x_i} - b)_+^2$ is the term which penalizes an incorrectly located point (an outlier). Now consider the two data sets with the LDB in Figure (9). All points in the two data sets are correctly classified except for the two points shown in bold. Those two points are considered as noise since they belong to one class but are located in the other class's half-space. Consider any correctly classified point from the positive class (class label $\mathrm{y} = +1$) which is not a support vector. That point $\boldsymbol{x}$ will always result in $\boldsymbol{w^T x} + b > 1$. Moreover, $\omega_i \in [0,1]$. This implies that $y_i - \boldsymbol{w}^T \boldsymbol{x_i} - b < -1$ and as a result, $\omega_i^{r+1} (y_i - \boldsymbol{w}^T \boldsymbol{x_i} - b)_+^2 = 0$. This means the objective function will not impose any penalty for a correctly classified point from the positive class. Now consider the same procedure but with a point from the negative class. This results in $\boldsymbol{w^T x} + b <$

-1 which implies that $y_i - \boldsymbol{w}^T \boldsymbol{x_i} - b > 1$, hence $\omega_i^{r+1}(y_i - \boldsymbol{w}^T \boldsymbol{x_i} - b)_+^2 > 0$. This means that the objective function will always penalize a correctly classified point from the negative class.

One can see that the same flaw is also found with noise points. For example, a noise point from the negative class located in the positive class half-space should be penalized. However, in this case $\boldsymbol{w^T x} + b > 1$ and $\omega_i^{r+1}(y_i - \boldsymbol{w}^T \boldsymbol{x_i} - b)_+^2 = 0$ which implies no penalties for a noise point from the negative class.



Class label
y = +1
Class label
y = -1
$\mathbf{w}^T\mathbf{x} + b > +1$
$\mathbf{w}^T\mathbf{x} + b < -1$
$x2$
$\mathbf{w}^T\mathbf{x} + b = +1$
$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b = -1$
$x1$

Figure 9: An illustration of two outliers imposing penalties

2.4 Problem Statement

Noise (or outliers) exist in data sets that are obtained from practical data analysis problems. The real challenge is in detecting and ignoring the noise while performing data analysis. Research areas like [21][22][23] require accurate results during the analysis. This highlights the need for robust classification models that can reduce the effect of noise in the development of the models and produce accurate end results.

From the literature, the SVM models [6][7][8] are sensitive to noise and deliver inaccurate results in the presence of outliers. Figures (10) and (11) show the effect of outliers on the decision boundary (separating line in green color) in a binary classification scenario.

In a robust SVM model, the following information is assumed to be given in the problem:

n Data points in m dimensions (features): $\boldsymbol{x}_i \in \mathbb{R}^m$

A label for each data point: $y_i = \{+1 \; if \; \boldsymbol{x}_i \in class \; 1; \; -1 \; if \; \boldsymbol{x}_i \in class \; 2\}$

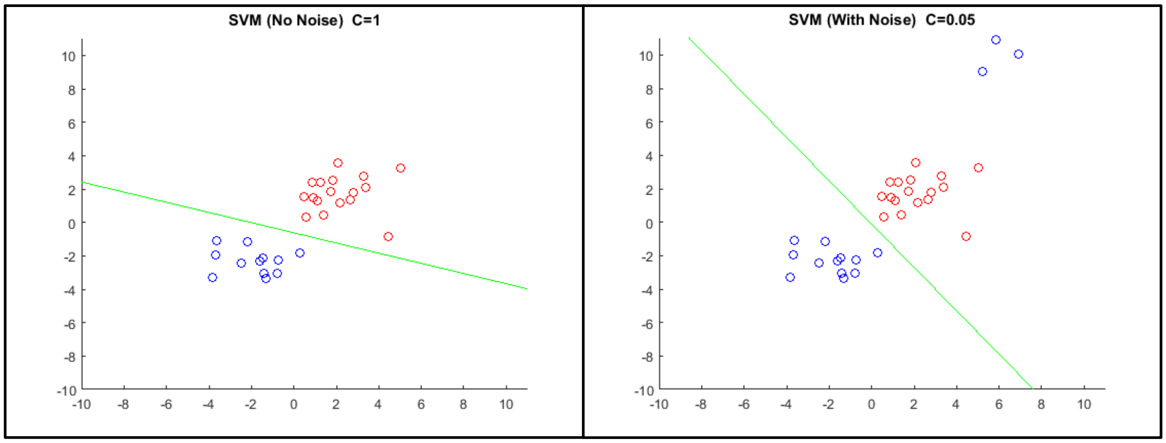


Figure 10: SVM best separating line before and after noise

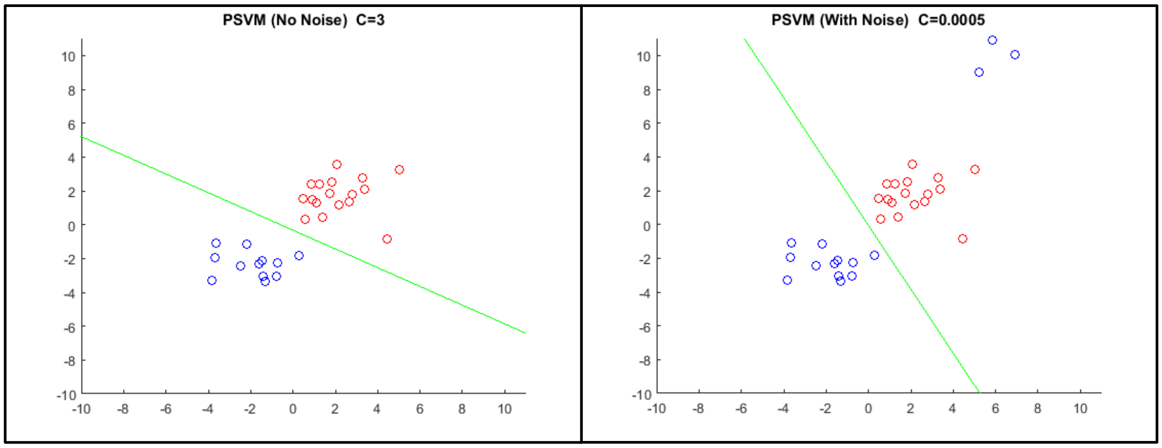


Figure 11: PSVM best separating line before and after noise

CHAPTER 3

ROBUST MATHEMATICAL MODELS

From the literature, it can be concluded that there are several approaches to induce robustness into the SVM models, including membership degrees and loss functions. In this thesis, three novel robust SVM models are proposed. In two of the proposed models the robustness is induced via a robust loss function, and in the third model the robustness is induced via the concept of membership function.

Two preferred characteristics in a loss function (ℓ) that makes SVM robust are boundedness and smoothness [16]. The first characteristic can be defined as $\lim_{\epsilon \to -\infty} \ell(\epsilon) = c_1$ and $\lim_{\epsilon \to +\infty} \ell(\epsilon) = c_2$, where $\epsilon \in \mathbb{R}$ is the absolute error such that $\epsilon = |y - f(\boldsymbol{x})|$, and $c_1, c_2 \in \mathbb{R}^+$. This characteristic in loss function is a key to induce robustness in the model. As the error ϵ increases beyond certain values, the loss function ℓ tends to remain constant, thus insensitive to the higher error values. The second characteristic (smoothness) implies that the loss function is continuously differentiable. As explained in Chapter 2, non-smoothness is a major concern in robust SVM models. Since it limits the optimization algorithms that can be used for the SVM models. The loss function in the three proposed models has these two characteristics, as it will be shown shortly.

The loss function in the proposed robust SVM models is derived from the correntropic loss function developed in [24]. The function is written as follows:

$$CLF = \beta\left(1 - e^{\frac{-z^2}{2\sigma^2}}\right), \tag{29}$$

where $\beta = \left(1 - e^{\frac{-1}{2\sigma^2}}\right)^{-1}$ is a normalization parameter, and $\sigma \in (0, \infty)$ is a scaling parameter which determines the shape of the function. Figure (12) shows the behavior of the function at different values of σ. It can be seen from the graph that the loss function increases (returns higher penalty) as ϵ increases or decreases. However, at certain values of ϵ and beyond, say $|\epsilon| \geq \epsilon^*$, the function becomes (and remains) insensitive to the values of ϵ. As an example, this means that the function values of a data point ($\epsilon = \epsilon^*$) and an outlier with a higher error ($\epsilon \gg \epsilon^*$) will be almost similar. This feature reduces the impact of high mis-classification error, hence reducing the function sensitivity towards outliers.

The correntropic function possesses some characteristics which make it very convenient and attractive to utilize in SVM models. Some of the most important characteristics are the following:

Continuity: the function is continuous over the domain

Differentiability (smoothness): the function is infinitely differentiable over the domain, which allows many optimization algorithms to be utilized in solving the SVM model.

Invexity: provides mathematical advantage while solving the SVM model.

Shape adjustability. The parameter σ can scale the function and tune the loss measuring capability.

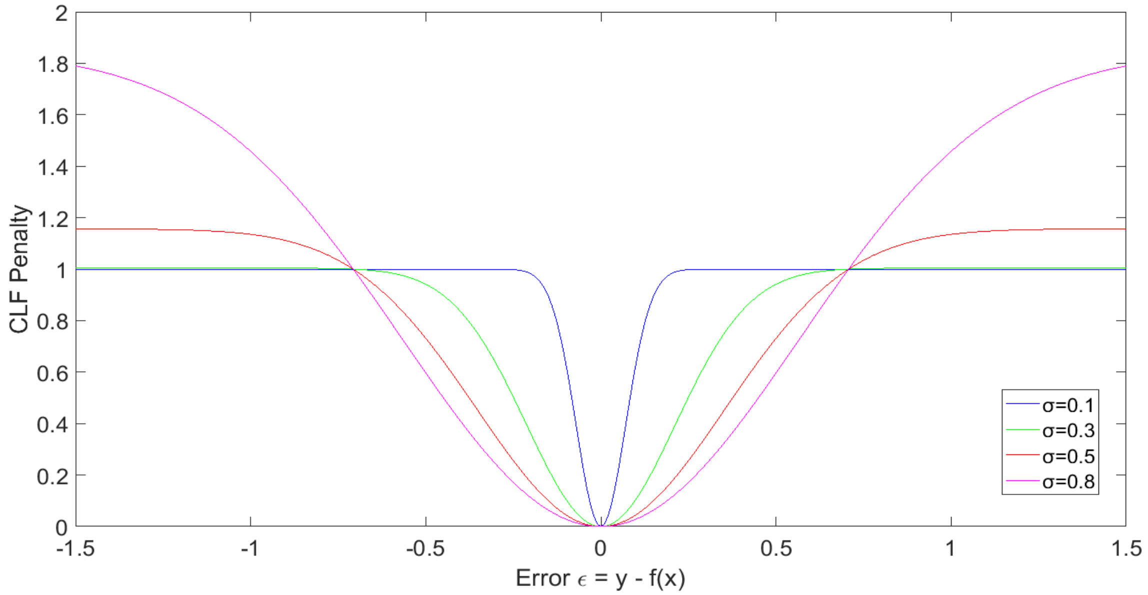


Figure 12: The correntropic function in terms of the absolute error

It is more convenient to start with the LDB models first and then use these models to develop the ones for NLDB. In this chapter, the three proposed models are presented for LDB. After that, the kernel trick and the representer theorem are discussed as key concepts for developing the NLDB models. Finally, the robust SVM models for NLDB are formulated using the generalized representer theorem.

3.1 Robust SVM Models for LDB

Typically, for SVM models, absolute error ϵ is not an apt measure of the misclassification error. For instance, if a point is correctly classified but is located far away from the decision boundary, then $f(\boldsymbol{x})$ will be very far from 1 (say the point belongs to class label 1). Thus, the absolute error will be high, i.e., $|1 - f(x)|$ will be high though the point is correctly classified. This indicates the need for an alternative way to calculate the error. In this thesis, the marginal error approach is considered.

Marginal error considers the sign matching instead of absolute difference. The marginal variable z is defined as the product of the decision boundary function and the associated label, $z = yf(\boldsymbol{x})$. Note that if the sign of the boundary function matches the class label, the product z will always be positive. On the other hand, if the two values mismatch then z will be negative.

The marginal variable by itself is not an error measure. To utilize z and develop what so called *marginal error* “ζ”, additional steps are required. From the SVM model, a point is correctly classified if it lies in the relevant half space or support hyper plane. This means a point between the decision boundary and the relevant hyper plane will be treated as a misclassified point. To encounter this idea in the calculation of the marginal error, we will have $\zeta = 1 - yf(\boldsymbol{x})$. Now we can easily realize that if $\zeta < 0$, then point $\boldsymbol{x}$ is correctly classified and if $\zeta > 0$ then $\boldsymbol{x}$ is misclassified. Since we need to penalize misclassified points only, we can then modify ζ to be as follows:

$$\zeta = \max\{0, 1 - yf(\boldsymbol{x})\} \tag{30}$$

The CLF can be then re-formulated using (30) to be as follows:

$$CLF = \beta\left(1 - e^{\frac{-(1-z)_+^2}{2\sigma^2}}\right), \tag{31}$$

where $(1-z)_+ = max\{0, 1-z\}$. It is important to note that the max function in the exponent precedes the square (power function), otherwise the modification will become meaningless. Figure (13) demonstrates the behavior of the CLF presented in Equation (31). The function returns zero as long as the point is correctly classified. Then the loss function starts to penalize as the marginal error from misclassification increases (towards the negative direction) until it settles on a certain value and becomes irresponsive to higher marginal errors.

In the following sections, the three proposed robust models are presented: Correntropic Support Vector Machine (C-SVM), Correntropic Proximal Support Vector Machine (C-PSVM), and Correntropic Membership Degree Support Vector Machine (CMD-SVM), respectively.

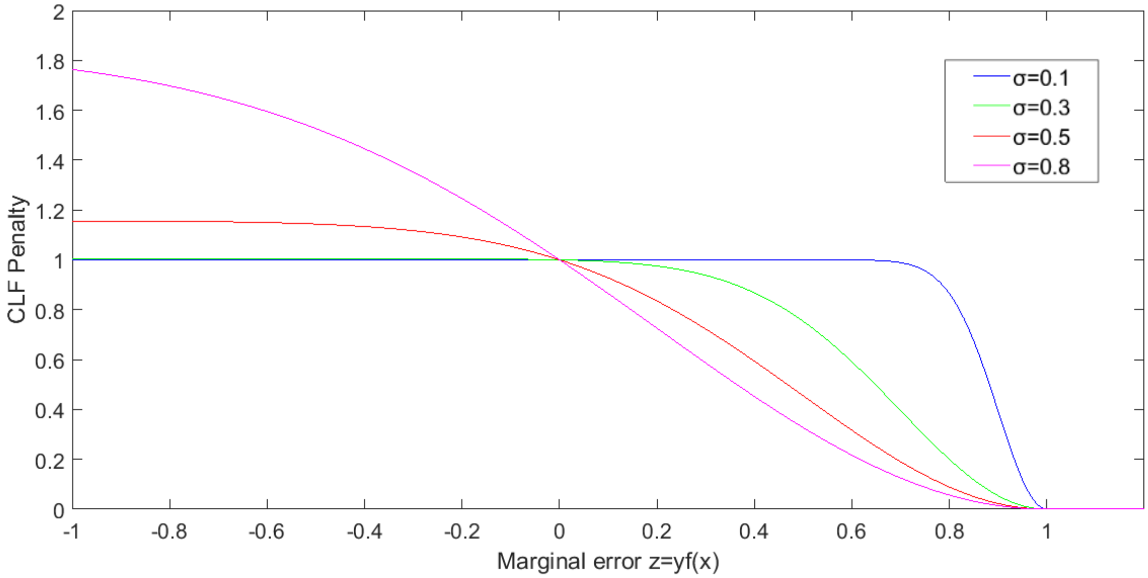


Figure 13: The CLF at different values of σ

3.1.1 Correntropic Support Vector Machine (C-SVM)

In C-SVM robust model, the correntropic function is incorporated into the objective function of the SVM model (9) to enhance its noise insensitivity.

The C-SVM model for a given λ and σ can be formulated as follows:

$$\text{min.:} \quad \lambda\|\boldsymbol{w}\|^2 + \frac{1}{n}\sum_i^n \left(1 - e^{\frac{-s_i^2}{\sigma^2}}\right) \tag{32a}$$

$$\text{s.t.:} \quad 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \leq s_i \quad \forall \; i \in 1, 2, \dots, n \tag{32b}$$

$$s_i \geq 0 \quad \forall \; i \in 1, 2, \dots, n \tag{32c}$$

The model also can be written in a compact form as follows:

$$\text{min.:} \quad \lambda\|\boldsymbol{w}\|^2 + \frac{1}{n}\sum_i^n \left(1 - e^{\frac{-\left(1 - y_i\left(w^T x_i + b\right)\right)_+^2}{\sigma^2}}\right) \tag{33}$$

The objective function consists of two terms. The first term $\|\boldsymbol{w}\|^2$ is the *regularization term* which tends to orient the decision boundary, so the maximum margin is obtained. The second term is the *loss function* which penalizes the incorrectly classified points but limits the penalty to a relatively low value for the outliers.

The loss function term imposes the robustness in the model. One can notice that minimizing these two terms under one objective function can be contradictory. In other words, a decision boundary that results in minimum loss function value may not achieve the maximum margin. The role of λ in the model is to set up the weight of the regularization term in order to tune the contribution of that term in the objective function with respect to the loss function.

3.1.2 Correntropic Proximal SVM (C-PSVM)

C-PSVM is the modification of PSVM using the correntropic function. The C-PSVM model for a given λ and σ is formulated as follows:

$$\text{min.:} \quad \lambda(\|\boldsymbol{w}\|^2 + b^2) + \frac{1}{n}\sum_i^n \left(1 - e^{\frac{-s_i^2}{\sigma^2}}\right) \tag{34a}$$

$$\text{s.t.:} \quad 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) = s_i \qquad \forall \ i \in 1, 2, \dots, n \tag{34b}$$

$$\text{min.:} \quad \lambda(\|\boldsymbol{w}\|^2 + b^2) + \frac{1}{n}\sum_i^n \left(1 - e^{\frac{-\left(1 - y_i\left(w^T x_i + b\right)\right)^2}{\sigma^2}}\right) \tag{35}$$

The model can also be written as:

As discussed in Section 2.1, the PSVM model outperforms the SVM model in solution time. It is highly likely that the C-PSVM will inherit the same property from the PSVM, at least to a certain extend. However, the only way to find out is by a rigorous mathematical proof or empirically by conducting numerical experiments. The C-PSVM performance with solution time will be demonstrated later in Section 5.4.

3.1.3 Correntropic Membership Degree SVM (CMD-SVM)

Based on the research directions from the literature survey, it can be seen that, designing the weights of the penalty term (the slack variable s_i) of the SVM model can reduce the effect of outliers. The key idea is that the weights associated to the outliers should have a very low value compared to the regular data points. However, the outliers are not known beforehand, and deciding the right value of weights is not straightforward.

The strong similarity measure characteristic found in the correntropic function will be utilized to identify the membership weights. The membership weights will determine the contribution (importance) of each data point in the model. The proposed CMD-SVM model at the t^{th} iteration and at given values for σ and λ is formulated as follows:

$$\text{min.}: \quad \lambda \|\boldsymbol{w}_t\|^2 + \frac{1}{n}\sum_i^n \alpha_i^{(t-1)} s_i \tag{36a}$$

$$\text{s.t.}: \quad 1 - y_i(\boldsymbol{w}_t^T \boldsymbol{x}_i + b_t) \leq s_i \quad \forall \ i \in 1, 2, \dots, n \tag{36b}$$

$$s_i \geq 0 \quad \forall \ i \in 1, 2, \dots, n \tag{36c}$$

where $\alpha_i^{(t)} = e^{\frac{-\left[1 - y_i\left(w_t^T x_i + b_t\right)\right]_+^2}{\sigma^2}}$ $\forall\, i,$ and the superscript (t) indicates the iteration number. At any iteration t, the CMD-SVM resembles the fuzzy membership model proposed in [9] with α_i being a constant which is obtained from the previous iteration. The model can be re-written in a compact form as follows:

$$\text{min.}: \quad \lambda \left\|\boldsymbol{w}^{(t)}\right\|^2 + \frac{1}{n}\sum_i^n \alpha_i^{(t-1)} \left[1 - y_i(\boldsymbol{w}_t^T \boldsymbol{x}_i + b)\right]_+ \tag{37}$$

In the following section extension of the proposed models in the nonlinear decision boundary is proposed.

3.2 Methods for NLDB

3.2.1 Dual Formulation

The formulations that have been discussed so far for C-SVM, C-PSVM, and CMD-SVM are primal model formulations, which intend to find a LDB. The objective function is minimized subject to a set of linear constraints with the decision variables being $\boldsymbol{w} \in \mathbb{R}^m$ and $b \in \mathbb{R}$. These two decision variables become the coefficients of the LDB function given in Equation (1). Therefore, the primal formulations as shown previously are not yet capable to produce NLDB. It is well known that the dual formulation of SVM model has a unique structure, which allows the model to exploit the kernel tricks to develop NLDB.

For the illustration purpose, the dual formulation for the SVM model (9) is derived using the Lagrange relaxation technique as:

$$\min_{\boldsymbol{w}, b} \max_{\boldsymbol{u}} \ \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{u}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + \sum_{i=1}^{n} u_i\big(1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)\big) \tag{38}$$

where $u_i \geq 0 \ \forall \ i$ are the dual variables. Since the SVM model given in (9) is convex, the strong duality holds and this implies the following:

$$\min_{\boldsymbol{w}, b} \max_{\boldsymbol{u}} \ \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{u}) \ = \ \max_{\boldsymbol{u}} \min_{\boldsymbol{w}, b} \ \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{u}) \tag{39}$$

The formulation in (38) can be rewritten using (39) as follows:

$$\max_{\boldsymbol{u}} \min_{\boldsymbol{w}, b} \ \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{u}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + \sum_{i=1}^{n} u_i\big(1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)\big) \tag{40}$$

For a given set of data points and their corresponding set of labels, the following stationarity conditions must satisfy at optimality for the inner minimization problem:

$$\nabla_{\boldsymbol{w}}\mathcal{L} = \boldsymbol{w} - \sum_{i=1}^{n} u_i y_i \boldsymbol{x}_i = 0 \;\; \rightarrow \;\; \boldsymbol{w} = \sum_{i=1}^{n} u_i y_i \boldsymbol{x}_i \tag{41a}$$

$$\nabla_{\boldsymbol{b}}\mathcal{L} = -\sum_{i=1}^{n} u_i y_i = 0 \;\; \rightarrow \;\; \sum_{i=1}^{n} u_i y_i = 0 \tag{41b}$$

Substituting (41a) and (41b) in (40), the problem can be written as:

$$\max_{\boldsymbol{u}} \mathcal{L}(\,\boldsymbol{u}) = \frac{1}{2}\sum_{i}^{n}\sum_{j}^{n} u_i u_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j + \sum_{i=1}^{n} u_i - \sum_{i=1}^{n} u_i y_i \boldsymbol{w}\boldsymbol{x}_i - \sum_{i=1}^{n} u_i y_i b \tag{42a}$$

$$\text{s.t.:} \qquad u_i \geq 0 \;\; \forall \;\; i \tag{42b}$$

$$\sum_{i=1}^{n} u_i y_i = 0 \tag{42c}$$

$$\boldsymbol{w} = \sum_{i=1}^{n} u_i y_i \boldsymbol{x}_i \tag{42d}$$

One can see that Equation (41b) implies the last term in the objective function (42a) to be zero. Additionally, $\sum_{i=1}^{n} u_i y_i \boldsymbol{w}\boldsymbol{x}_i = \sum_{i}^{n}\sum_{j}^{n} u_i u_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j$, which can be easily shown using constraint (42d). Finally, constraint (42d) will no longer be needed since $\boldsymbol{w}$ will disappear from the new formulation after substituting all $\boldsymbol{w}$'s with (42d). This implies that the new formulation can be reduced as:

$$\max_{\boldsymbol{u}} \mathcal{L}(\boldsymbol{u}) = \sum_{i=1}^{n} u_i - \frac{1}{2}\sum_{i}^{n}\sum_{j}^{n} u_i u_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j \tag{43a}$$

s.t. :

$$u_i \geq 0 \ \forall \ i \tag{43b}$$

$$\sum_{i=1}^{n} u_i y_i = 0 \tag{43c}$$

So far, the dual formulation in (43) will provide a LDB in terms of $\boldsymbol{w}$ and b. The LDB can be obtained from $\boldsymbol{u}^*$ as follows: Typically, $\boldsymbol{u}^*$ is normally sparse. Hence the values of $\boldsymbol{u}$ such that $u_i > 0$ plays a critical role in identifying the LDB. If $u_i > 0$ then the associated data point $\boldsymbol{x}_i$ will be the "support vector" of the LDB. Support vectors are the data points that lie on the support hyperplanes and as a result, dictate the position and the slope of the LDB. Now to find $\boldsymbol{w}^*$, we can refer to (42d) which yields to $\boldsymbol{w}^* = \sum_{i=1}^{n} u_i^* y_i \boldsymbol{x}_i$. b^* can be computed by solving any of the two support hyperplanes' equation for b using a support vector that lies on the same support hyperplane. For example, the equation of the positive support hyperplane is $\boldsymbol{w}^T \boldsymbol{x}_i + b = +1$ and a support vector $\hat{\boldsymbol{x}}_q$ on that hyperplane with $u_q > 0$ will lead to $b^* = \mathbf{1} - \boldsymbol{w}^{*\boldsymbol{T}} \hat{\boldsymbol{x}}_q$.

The discussion so far has been related to the LDB. In the primal model, the decision boundary is defined so that the margin is maximum, then the support hyperplanes identify the support vectors. In the dual model, the support vectors are identified first, then the orientation of the decision boundary is defined such that the support hyperplanes lie on the support vectors [3].

Now, to identify the NLDBs, typically the kernel tricks are used. The key idea of using kernels is to project the current data into high dimension Hilbert space, such that the low dimension NLDB converts to LDB in the higher dimensions. The inner product $\boldsymbol{x}_i^T\boldsymbol{x}_j$ available in the objective function of the dual formulation serves as a gate for the usage of the kernel tricks. The following sub-section explains the utilization of the inner product to manipulate the feature space and develop nonlinear separation for binary classification.

3.2.2 Feature Space and Kernel Trick

It is critical to emphasize that NLDB may not be represented by a non-linear function. Generally, the NLDB in the lower dimension is represented by a corresponding LDB in the higher dimension. Therefore, the key is to transform the linearly inseparable classes into a space where they can be linearly separable. This can be accomplished by adding a new dimension (or multiple dimensions) to the feature space. For example, in Figure (14), the given set of data points $\boldsymbol{x} \in \mathbb{R}^2$ are not linearly separable. Thus, a third dimension in the feature space can be created in terms of the two existing dimensions using a mapping function $\Phi(\boldsymbol{x})$. This additional dimension will modify the data structure and allow for a LDB to exist and separate these two classes. This concept is demonstrated in Figures (15) and (16).

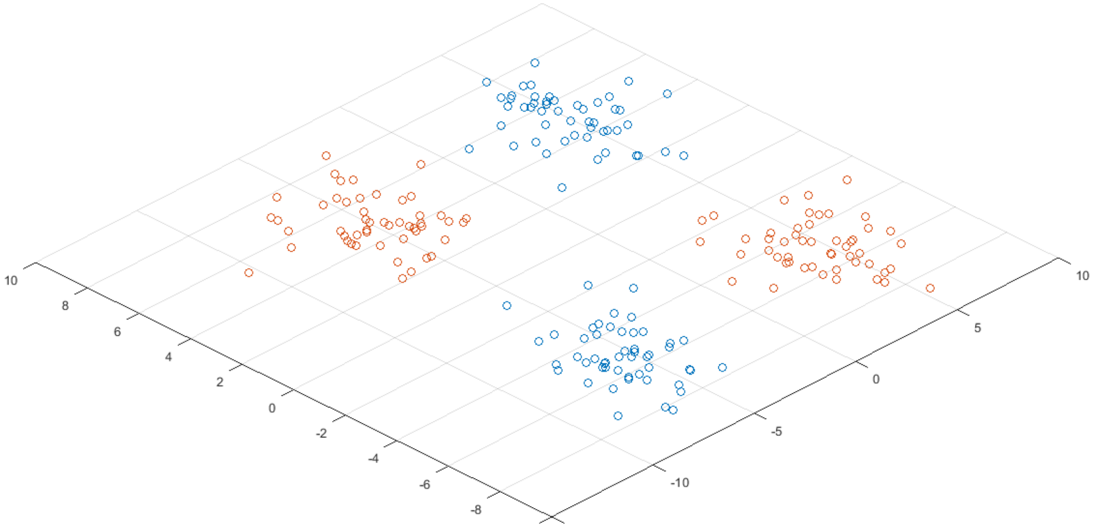


Figure 14: Data points for two classes are created in 2D (m=2)

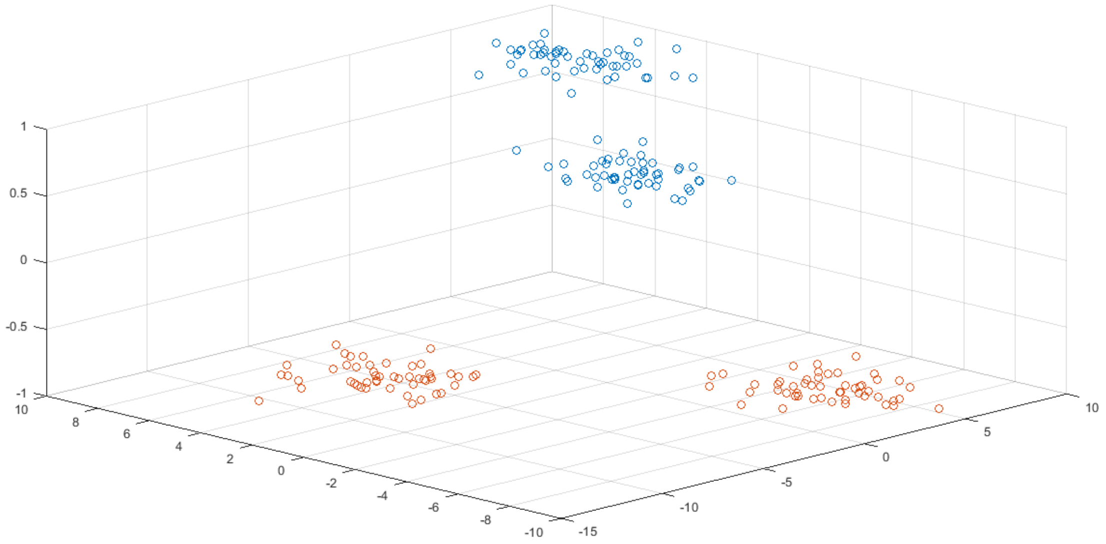


Figure 15: Adding one space dimension (the vertical z-axis) and projecting the blue class along that new axis

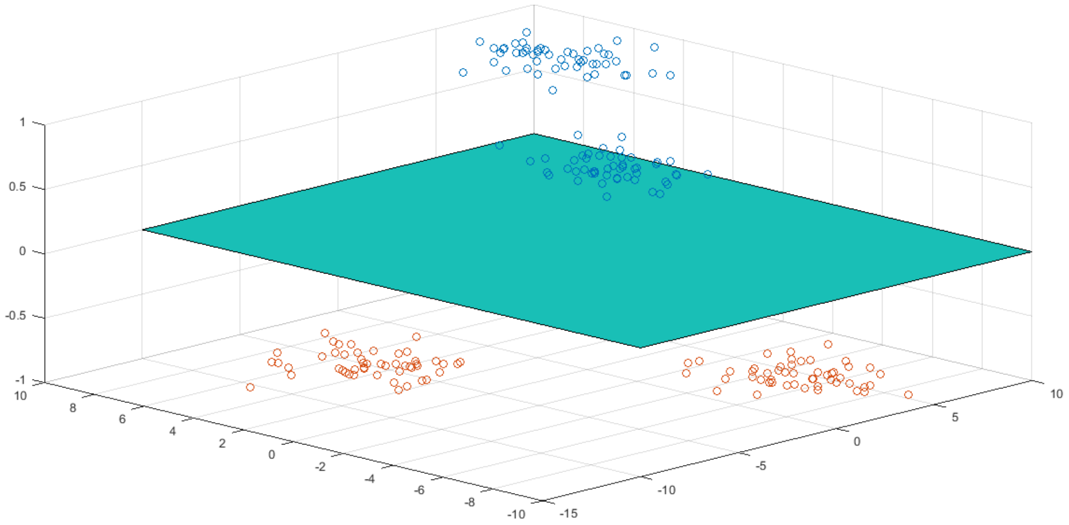


Figure 16: The mapping from 2D to 3D allowed the two classes to be linearly separable

Based on the structure of the data sets, it may be necessary to add multiple dimensions to achieve the linear separability. This process of modifying the feature space can be difficult or sometimes inefficient. Moreover, the overall computation time added by the new dimension is often significant. For instance, the 2nd degree polynomial mapping for $\boldsymbol{x} \in \mathbb{R}^2$ can be written as:

$\Phi(\boldsymbol{x}) = [x_1^2\,, \quad x_2^2\,, \quad \sqrt{2}\,x_1\,, \quad \sqrt{2}\,x_2\,, \quad \sqrt{2}\,x_1 x_2\,, \quad 1]$, which transforms the points from $\mathbb{R}^2 \qquad \mathbb{R}^6$. The dot product of two points will be as follows:

$$\Phi(\boldsymbol{x_i})^{\mathrm{T}}\Phi(\boldsymbol{x_j}) = x_{1i}^2 x_{1j}^2 + \; x_{2i}^2 x_{2j}^2 + 2x_{1i}x_{1j} + 2x_{2i}x_{2j} + 2x_{1i}x_{1j}x_{2i}x_{2j} + 1 \qquad (45)$$

This shows the computation involved in a simple second order polynomial mapping of $\boldsymbol{x} \in \mathbb{R}^2$ only. To avoid the bulky computations, kernel trick is used. The trick is to obtain the required high dimensional information without actually visiting the high dimensional space. This trick is implemented by using a “kernel function” $K\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$.

Generally, the kernel function is a transformation function that obtains the higher dimensional information without visiting that space. The information of the dot product in (45), can be obtained using the following kernel function:

$$\Phi(\boldsymbol{x_i})^{\mathrm{T}}\Phi(\boldsymbol{x_j}) = \left(\boldsymbol{1} + x_{1i}x_{1j} + x_{2i}x_{2j}\right)^2 = \left(1 + \boldsymbol{x}_i^T \boldsymbol{x}_j\right)^2 = K\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \qquad (46)$$

Although (45) and (46) provide the same result, they are dissimilar with regard to the computation effort. It is clear that much time is required to execute (45), whereas the only significant computation in (46) is the dot product. Moreover, the data points dimension m has a tremendous impact on computing (45). On the other hand, m in (46) affects the computation of only the dot product $\boldsymbol{x}_i^T \boldsymbol{x}_j$. In most cases where multiple dimensions mapping is needed or when the data points dimension is high, an explicit computation like the one in (45) are impractical. Therefore, kernel functions like the one shown in (46) provides a huge advantage.

Now the advantage of the dual formulation for the SVM can be realized. Based on the above discussion, the data points are mapped as follows $\Phi\colon \boldsymbol{x} \rightarrow \Phi(\boldsymbol{x})$. Moreover, the kernel function of any two points is the dot product of the mapping functions of these two points. This implies $\boldsymbol{x}_i^T \boldsymbol{x}_j \rightarrow \Phi(\boldsymbol{x_i})^{\mathrm{T}}\, \Phi(\boldsymbol{x_j}) = K\langle \boldsymbol{x_i}, \boldsymbol{x_j}\rangle$. As a result, the dual problem of the SVM can be reformulated by replacing the dot product $\boldsymbol{x}_i^T \boldsymbol{x}_j$ with $K\langle \boldsymbol{x_i}, \boldsymbol{x_j}\rangle$.

The kernelized dual problem of the SVM is solved for $\boldsymbol{u}$**,** similar to solving the dual problem explained in Sub-section 3.2.1. Nevertheless, the two models yield to two different solutions of $\boldsymbol{u}^*$. The kernelized dual formulation will identify the support vectors ($\boldsymbol{u_i}$ >1) in a way that a NLDB can be achieved on the primal domain.

It is interesting to know that some kernel functions perform the transformation in infinite dimension space, yet the problem dimension remains the same. This shows the power of the kernal trick and the convenience of using it for classification problems with large number of features, where transformation into very high dimension spaces is required.

There are many transformation functions that can be used in the process to develop a NLDB. However, not every transformation function is a kernel function. For a transformation function to be classified as kernel function, the function needs to satisfy Mercer condition. The Mercer theorem is summarized as follows [25]:

If $K(\boldsymbol{a}, \boldsymbol{b})$ satisfies Mercer's condition:

$$\int_a \int_b K(\boldsymbol{a}, \boldsymbol{b}) g(\boldsymbol{a}) g(\boldsymbol{b})\, da\, db \geq 0 \;\; \forall \text{ data sets } \{x: g^T K g \geq 0\}$$

$$\text{then } K(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{\Phi}(\boldsymbol{a}) \cdot \boldsymbol{\Phi}(\boldsymbol{b}) \text{ for some } \Phi(x)$$

where K is a kernel function, $\Phi(x)$ is a mapping function, $\boldsymbol{a}$ and $\boldsymbol{b}$ are data points in $\mathbb{R}^n$. As a result of satisfying the above Mercer's condition, the kernel function will possess the finitely positive semi-definite property. This property privileges the kernel function to have that unique performance in space transformation and similarity measure [25]. Some examples of well-known kernel functions are the linear kernel, homogeneous polynomial kernel, and Gaussian kernel (also known as radial basis function).

3.2.3 Generalized Representer Theorem

Sub-sections 3.2.1 and 3.2.2 discuss the key concepts to construct an SVM model for NLDB. Developing the dual SVM models eliminate the primal variables $\boldsymbol{w}$ and b and introduces a new term $\boldsymbol{x}_i^T \boldsymbol{x}_j$, which then can be kernelized as follows:

$$\boldsymbol{x}_i^T \boldsymbol{x}_j = K\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \Phi(\boldsymbol{x}_i)^{\mathrm{T}} \Phi\left(\boldsymbol{x}_j\right) \tag{47}$$

Depending on the primal formulations of the proposed robust models, deriving the dual in terms of only the Lagrange multipliers (eliminating $\boldsymbol{w}$ and b from the formulation), may not be achievable. Moreover, the dot product of the training points $\boldsymbol{x}_i^T \boldsymbol{x}_j$ may not appear in the objective function of the dual formulation, which is the main motivation for deriving the dual. It is clear that utilizing the dual formulation to apply the kernel trick can be very challenging. This challenge can be overcome by utilizing the Generalized Representer Theorem (GRT).

The GRT allows the primal model to be kernelized without the need for the dual formulation. The representer theorem is generalized in [26] to accommodate a wider class of regularizers and risk (loss) functions. The theorem states that minimizing a regularizer g and a loss function l can be represented in a kernelized form in the primal formulation if certain conditions are satisfied. The theorem is summarized as follows:

Given a nonempty set of training samples $(\boldsymbol{x_1},\, y_1),\ \cdots,(\boldsymbol{x_n},\, y_n) \in \mathbb{R}^m \times \mathbb{R}$, a class of functions $\mathcal{F} = \{f \in \mathbb{R}^m | f(\boldsymbol{x}) = \sum_{j=1}^{n} u_j k(\boldsymbol{x},\, \boldsymbol{x}_j),\, u_j \in \mathbb{R}, \boldsymbol{x_j} \in \mathbb{R}^{\boldsymbol{m}},\, \|f\| < \infty\}$, an arbitrary loss function $l: (\mathbb{R} \times\ \mathbb{R}^m) \to\ \mathbb{R} \cup \{\infty\}$, a function $g \to\ \mathbb{R}$, and a set of M real-valued function $\{\psi_p(\boldsymbol{x})\}_{p=1}^{M}$, then a function $\hat{f} \coloneqq f + h$, with $f \in \mathcal{F}$ and $h \in \operatorname{span}\{\psi_p\}$, that is minimizing the objective function:

$$H = g(\|f\|) + l\left(\left(y_1,\, \hat{f}(\boldsymbol{x_1})\right),\ \cdots,\ \left(y_n, \hat{f}(\boldsymbol{x_n})\right)\right), \tag{48}$$

can be represented in the so-called “representation form” (or kernelized form) as:

$$\hat{f}(\boldsymbol{x}) = \sum_{j=1}^{n} u_j k(\boldsymbol{x}, \boldsymbol{x}_j) + \sum_{p=1}^{M} \beta_p \psi_p(\boldsymbol{x})\,, \tag{49}$$

where $k \in \mathbb{R}^m$ is a kernel function and $\beta_p \in \mathbb{R}\ \forall\, p = 1,\ \cdots,\ M$ are unique coefficients, only if the following conditions are satisfied:

1) The kernel function k is a real-valued positive definite on $\mathbb{R}^m$.

2) Function $g \to\ \mathbb{R}$ is strictly monotonically increasing on $[0,\ \infty)$.

3) The set of functions $\{\psi_p\}_{p=1}^{M}$ on $\boldsymbol{X}$ represented by the $n \times M$ matrix $(\psi_p(\boldsymbol{x}_i))_{ip}$ has a rank M.

The theorem states that if a primal model is in GRT form, and the three conditions are satisfied, then the proposed decision boundary function $\hat{f}(\boldsymbol{x})$ is a valid representation for the kernelized model.

3.3 Robust SVM Models for NLDB

The motivation behind using the GRT is the simplicity of constructing the kernelized model without the need to go through the detailed mathematics of deriving the dual. We also explained previously that the dual formulation needs to meet certain criteria in order for the kernelization to be applicable. In other words, the dual formulation does not guarantee the appearance of the dot product $\boldsymbol{x}_i^T \boldsymbol{x}_j$ and the disappearance of $\boldsymbol{w}$ and b. This indicates that utilizing the GRT is easier once the model satisfies the required six conditions. In this thesis, the GRT will be used to develop the robust SVM models (C-SVM, C-PSVM, and CMD-SVM) for NLDB. In each of the following subsections, the theorem is proved to be applicable to the model, followed by the construction of the kernelized robust model.

3.3.1 Correntropic Support Vector Machine (C-SVM)

The C-SVM model for LDB formulated in (9) can be reformulated in a compact form as follows:

$$\text{min.}: \quad \lambda\|\boldsymbol{w}\|^2 + \frac{1}{n}\sum_i^n \left(1 - e^{\frac{-\left(1-y_i\left(\boldsymbol{w}^T x_i+b\right)\right)_+^2}{\sigma^2}}\right) \tag{50}$$

The compact formulation is more convenient when dealing with the GRT since the theorem is built to deal with unconstrained models. Now we need to build the kernelized objective function as per the format stated in the representer theorem in (48) which is:

$$g(\|f\|) + c\left(\left(y_1, \hat{f}(\boldsymbol{x_1})\right), \cdots, \left(y_m, \hat{f}(\boldsymbol{x_m})\right)\right) \tag{48}$$

where $\hat{f} := f + h$ is the decision boundary function formulated in the kernelized form as:

$$\hat{f}(\boldsymbol{x}) = \sum_{j=1}^{n} u_j k\left(\boldsymbol{x}, \boldsymbol{x}_j\right) + \sum_{p=1}^{M} \beta_p \psi_p(\boldsymbol{x}), \tag{49}$$

The function f can be simply defined as $f(\boldsymbol{x}) = \sum_{j=1}^{n} u_j k\left(\boldsymbol{x}, \boldsymbol{x}_j\right) = h$, where $f \in \mathcal{F}$ is clearly seen. Furthermore, by selecting $M = 1$, we have $\left\{\psi_p\right\}_{p=1}^{M} = \boldsymbol{\psi} \in \mathbb{R}^n$. Note that ψ is a function of the data points $\boldsymbol{x}$ (i.e. $\psi(\boldsymbol{x}) \in \mathbb{R}$). However, we can define ψ to be a constant function such that $\psi(\boldsymbol{x_i}) = b \;\forall\, i$. By having $\beta_p = \beta = 1$, the decision boundary function in Equation (49) can be redefined as follows:

$$\hat{f}(\boldsymbol{x}) = \sum_{j=1}^{n} u_j k\left(\boldsymbol{x}, \boldsymbol{x_j}\right) + b \tag{51}$$

This implies that the decision boundary in the primal form (i.e. $\boldsymbol{w}^T \boldsymbol{x}_i + b = 0$) is kernelized into the kernelized form as $\sum_{j=1}^{n} u_j k\left(\boldsymbol{x}_i, \boldsymbol{x_j}\right) + b = 0$.

Now consider the following:

a) The loss function $l = \frac{1}{n}\sum_i^n \left(1 - e^{\frac{-\left[1- y_i\left(w^T x_i + b\right)\right]_+^2}{\sigma^2}}\right)$ can be formulated in the kernelized form using Equation (51). The kernelized loss function is written as: $l = \frac{1}{n}\sum_i^n \left(1 - e^{\frac{-\left[1- y_i\left(\sum_{j=1}^n u_j k\left(\boldsymbol{x_i}, \boldsymbol{x_j}\right)+b\right)\right]_+^2}{\sigma^2}}\right).$

b) The regularization function g can be defined in our model to be the square function. This implies that $g(\|\cdot\|) = \lambda(\|\cdot\|)^2$.

By incorporating points (a) and (b), it is easy to show that

$$g(\|f\|) + c\left(\left(y_1, \hat{f}(\boldsymbol{x_1})\right), \cdots, \left(y_m, \hat{f}(\boldsymbol{x_m})\right)\right)$$

$$= g\left(\left\|\sum_{j=1}^{n} u_j k\left(\boldsymbol{x}, \boldsymbol{x_j}\right)\right\|\right) + c\left(\left(y_1, \sum_{j=1}^{n} u_j k\left(\boldsymbol{x_1}, \boldsymbol{x_j}\right) + b\right), \cdots, \left(y_m, \sum_{j=1}^{n} u_j k\left(\boldsymbol{x_m}, \boldsymbol{x_j}\right) + b\right)\right)$$

$$= \lambda \sum_i \sum_j u_i u_j k\left(\boldsymbol{x_i}, \boldsymbol{x_j}\right) + \frac{1}{n}\sum_i^n \left(1 - e^{\frac{-\left[1- y_i\left(\sum_{j=1}^n u_j k\left(\boldsymbol{x_i}, \boldsymbol{x_j}\right)+b\right)\right]_+^2}{\sigma^2}}\right)$$

This concludes that the objective function is constructed as per the theorem with the loss function l written in terms of $\hat{f}$. Nevertheless, the three conditions still need to be satisfied before we conclude the validity of our kernelization. The three conditions are verified as follows:

1) *The kernel function k is a real-valued positive definite on* $\mathbb{R}^m$: in this thesis, the Radial Basis Function (RBF) is used for NLDB models. RBF is a kernel function

satisfying Mercer's condition [25]. Therefore, the function is a real-valued positive definite function over its domain.

2) *Function* $g \rightarrow \mathbb{R}$ *is strictly monotonically increasing on* $[0,\infty)$: the function g was chosen to be a parabolic function centered over the origin. It is well known that g in this case is strictly convex with a global minimum at the origin (i.e. $\|\boldsymbol{w}\| = 0$). Therefore, g is strictly monotonically increasing on $[0,\infty)$.

3) *The set of functions* $\{\psi_p\}_{p=1}^{M}$ *on* $\boldsymbol{X}$ *represented by the* $n \times M$ *matrix* $(\psi_p(\boldsymbol{x}_i))_{ip}$ *has a rank* M: since M is defined as $M = 1$ in this model, ψ becomes a vector of size n, which means this condition will always be satisfied.

This implies that $\hat{f}$ defined in (51) is a valid kernelized representation of the primal decision boundary function and the C-SVM model for NLDB can be represented as:

$$\text{min.}: \quad \lambda \sum_i \sum_j u_i u_j k(\boldsymbol{x_i}, \boldsymbol{x_j}) + \frac{1}{n} \sum_i^n \left(1 - e^{\frac{-s_i^2}{\sigma^2}}\right) \tag{52a}$$

$$\text{s.t.}: \qquad 1 - y_i \left(\sum_{j=1}^{n} u_j k(\boldsymbol{x_i}, \boldsymbol{x_j}) + b \right) \leq s_i \qquad \forall \; i \in 1, 2, \dots, n \tag{52b}$$

$$s_i \geq 0 \qquad \forall \; i \in 1, 2, \dots, n \tag{52c}$$

where $k(\boldsymbol{x_i}, \boldsymbol{x_j}) = e^{-\gamma \|x_i - x_j\|^2}, \gamma \in \mathbb{R}^+$.

3.3.2 Correntropic Proximal SVM (C-PSVM)

The C-PSVM model for LDB has a lot of similarities compared to the LDB C-SVM, hence the method to derive the C-PSVM for NLDB is not expected to be very different from that of C-SVM. The C-PSVM for LDB is reformulated into a compact form as:

$$\text{min.:} \quad \lambda(\|\boldsymbol{w}\|^2 + b^2) + \frac{1}{n}\sum_i^n \left(1 - e^{\frac{-\left(1- y_i\left(w^T x_i+b\right)\right)^2}{\sigma^2}}\right) \tag{53}$$

Following the same procedure before applying the theorem, the objective function needs to be written in the standard form proposed by the theorem and that is:

$$g(\|f\|) + c\left(\left(y_1, \hat{f}(\boldsymbol{x_1})\right), \cdots, \left(y_n, \hat{f}(\boldsymbol{x_n})\right)\right) \tag{48}$$

The C-PSVM in (10) has the term b^2 appearing in the objective function, which slightly complicates the implementation of the theorem. One way to come over this obstacle is to reformulate (53) such that b^2 disappears but the general structure remains the same. This can be achieved by defining a new variable $\widehat{\boldsymbol{w}}$ and a new parameter $\widehat{\boldsymbol{x}}_i$. The variables $\boldsymbol{w}$ and b can be combined in the new augmented variable $\widehat{\boldsymbol{w}}$ such as $\widehat{\boldsymbol{w}} = \begin{bmatrix} \boldsymbol{w} \\ b \end{bmatrix}$. Note that $\boldsymbol{w}$ and b will be implicit in the new formulation and the dimension of the replacement variable will be $\widehat{\boldsymbol{w}} \in \mathbb{R}^{m+1}$. This new dimension of $\widehat{\boldsymbol{w}}$ will disable the dot product because we still have $\boldsymbol{x_i} \in \mathbb{R}^{\boldsymbol{m}}$. This issue can be resolved by assigning $\widehat{\boldsymbol{x}}_i = \begin{bmatrix} \boldsymbol{x_i} \\ 1 \end{bmatrix}$. The C-PSVM for LDB (53) can then be re-written as:

$$\text{min.:} \quad \lambda\|\widehat{\boldsymbol{w}}\|^2 + \frac{1}{n}\sum_i^n \left(1 - e^{\frac{-\left(1- y_i \widehat{w}^T \widehat{x}_i\right)^2}{\sigma^2}}\right) \tag{54}$$

One can easily see that (53) and (54) are identical by substituting $\widehat{\boldsymbol{w}}$ and $\widehat{\boldsymbol{x}}_i$ and writing the model in terms of $\boldsymbol{w}$, b, and $\boldsymbol{x_i}$ explicitly. It can be also clearly seen that (54) is in the theorem proposed form. However, because the bias term does not appear in the exponential term in the loss function, we will have the decision boundary function as:

$$\hat{f} = f + h = f + 0 = f \tag{55}$$

Equation (55) implies that $\sum_{p=1}^{M} \beta_p \psi_p(\boldsymbol{x}) = \beta\psi(\boldsymbol{x}) = 0$. As a result, the kernelized form of f as well as $\hat{f}$ is formulated as:

$$f(\widehat{\boldsymbol{x}}) = \hat{f}(\widehat{\boldsymbol{x}}) = \sum_{j=1}^{n} u_j k(\widehat{\boldsymbol{x}}, \widehat{\boldsymbol{x}}_{\boldsymbol{j}}) \tag{56}$$

Furthermore, the regularization function is defined as $g(\|\cdot\|) = \lambda(\|\cdot\|)^2$ and the loss function in the kernelized form is defined as $l = \frac{1}{n}\sum_i^n \left(1 - e^{\frac{-\left(1 - y_i \sum_{j=1}^{n} u_j k(\widehat{\boldsymbol{x}}_{\boldsymbol{i}}, \widehat{\boldsymbol{x}}_{\boldsymbol{j}})\right)^2}{\sigma^2}}\right)$. By verifying the three conditions of the theorem and referring to the GRT implementation for the C-SVM model, it can be concluded that the three conditions are satisfied. The C-PSVM model for NLDB is then formulated as:

$$\text{min.:} \quad \lambda \sum_i \sum_j u_i u_j k(\widehat{\boldsymbol{x}}_{\boldsymbol{i}}, \widehat{\boldsymbol{x}}_{\boldsymbol{j}}) + \frac{1}{n}\sum_i^n \left(1 - e^{\frac{-s_i^2}{\sigma^2}}\right) \tag{57a}$$

$$\text{s.t.:} \qquad 1 - y_i \sum_{j=1}^{n} u_j k(\widehat{\boldsymbol{x}}_{\boldsymbol{i}}, \widehat{\boldsymbol{x}}_{\boldsymbol{j}}) = s_i \qquad \forall\ i \in 1, 2, \dots, n \tag{57b}$$

where $\widehat{\boldsymbol{x}}_i = \begin{bmatrix} \boldsymbol{x}_{\boldsymbol{i}} \\ 1 \end{bmatrix}$, $k(\widehat{\boldsymbol{x}}_{\boldsymbol{i}}, \widehat{\boldsymbol{x}}_{\boldsymbol{j}}) = e^{-\gamma\|\widehat{x}_i - \widehat{x}_j\|^2}, \gamma \in \mathbb{R}^+$.

3.3.3 Correntropic Membership Degree SVM (CMD-SVM)

Applying the GRT for CMD-SVM will be exactly the same as in the C-SVM except that the loss function is slightly different. Recall the CMD-SVM model for LDB written in the compact form as:

$$\text{min.}: \quad \lambda \left\| \boldsymbol{w}^{(t)} \right\|^2 + \frac{1}{n} \textstyle\sum_i^n \alpha_i^{(t-1)} \left[1 - y_i \left({\boldsymbol{w}^{(t)}}^T \boldsymbol{x}_i + b^{(t)} \right) \right]_+ \tag{58}$$

where $\alpha_i^{(t)} = e^{\frac{-\left[1 - y_i \left({w^{(t)}}^T x_i + b^{(t)} \right) \right]_+^2}{\sigma^2}}$ $\forall\, i,$ and t indicates the iteration number.

The kernelized form of the decision boundary $\hat{f} \coloneqq f + h$ is formulated as:

$$\hat{f}(\boldsymbol{x}) = \textstyle\sum_{j=1}^n u_j k\left(\boldsymbol{x}, \boldsymbol{x_j}\right) + b, \tag{49}$$

With $f = \sum_{j=1}^n u_j k\left(\boldsymbol{x}, \boldsymbol{x_j}\right)$ and $h = b$. As a result, the kernelized form of the loss function is written as $l = \frac{1}{n} \sum_i^n \alpha_i^{(t-1)} \left[1 - y_i \left(\sum_{j=1}^n u_j k\left(\boldsymbol{x_i}, \boldsymbol{x_j}\right) + b\right)\right]$. The regularization function is the same for all three models which is $g(\cdot) = (\cdot)^2$. The three conditions of the theorem can be verified the similar to the C-SVM, and it can be concluded that all conditions are satisfied. The CMD-SVM can be kernelized for NLDB as follows:

$$\text{min.}: \quad \lambda \textstyle\sum_i \sum_j u_i u_j k\left(\boldsymbol{x_i}, \boldsymbol{x_j}\right) + \frac{1}{n} \sum_i^n \alpha_i^{(t-1)} s_i \tag{59a}$$

$$\text{s.t.}: \quad 1 - y_i \left(\sum_{j=1}^n u_j k\left(\boldsymbol{x_i}, \boldsymbol{x_j}\right) + b \right) \leq s_i \qquad \forall \; i \in 1, 2, \dots, n \tag{59b}$$

$$s_i \geq \ 0 \qquad \forall \ i \in 1, 2, \dots, n \tag{59c}$$

where $\alpha_i^{(t)} = e^{\frac{-\left[1-y_i\left(\sum_{j=1}^{n} u_j^{(t)} k\left(x_i, x_j\right)+b^{(t)}\right)\right]_+^2}{\sigma^2}}$, $k\left(\boldsymbol{x_i}, \boldsymbol{x_j}\right) = e^{-\gamma\|x_i - x_j\|^2}$, λ and $\gamma \in \mathbb{R}^+$.

The three models formulated in (52), (57), and (59) are the kernelized formulation of the robust models developed in Section 3.1. These kernelized models are solved to find $\boldsymbol{u}^*$ and b which are then used to build a NLDB. The nonlinear classification is conducted numerically using the associated kernelized rule function.

The proposed robust models for LDB and NLDB embed the admired characteristics of the CLF, which grant them unique behavior. In next chapter, an iterative solution methodology is proposed, which exploits the potential capability of these proposed robust models.

CHAPTER 4

SOLUTION METHODOLOGY

The solution methodology is a critical stage of any optimization problem. It demonstrates the value of the proposed model, thus highlights the motivation behind using this model instead of others. The solution methodology is usually developed based on the problem (or the model) structure (convex/non-convex, smooth etc.). A good methodology tends to exploit some unique characteristics of the model in order to produce better solutions or improve solution times.

The solution methodology for each of the three models (C-SVM, C-PSVM, and CMD-SVM) will follow the same strategy for both cases (LDB and NLDB). Therefore, the C-SVM model is used for illustrations and explanation of the solution methodology. The illustrated ideas will then be generalized for the other remaining models and cases. The CMD-SVM model will have a slightly different solution algorithm compared to the other two models, because the model is based on the membership degrees (rather than the loss functions as in the other two models). The methodology that is specifically developed for the CMD-SVM model is discussed at the end of this chapter.

4.1 Model Convexity and Role of (σ)

The two models, C-SVM and C-PSVM, are non-convex optimization models for any given σ. This non-convexity is imposed by the non-convex correntropic loss function in the objective function. The parameter σ, as a tuning parameter for the loss function, determines the shape and geometry of the function. The effect of σ is what makes the function unique. For relatively low values of σ, the function demonstrates high non-convex behavior globally. However, the function tends to be locally pseudoconvex until σ becomes significantly large, then the function becomes strictly convex [27]. Figure (17) illustrates the graph of a C-SVM model (in the compact form Equation (33)) during the pseudoconvex stage. The figure also names some critical areas to simplify the discussion in the following sections.

Pseudoconvex functions have good characteristics and they are the preferred in the field of optimization after the convex functions [28]. Since the regularization term in the objective function $\boldsymbol{u}^T\boldsymbol{k}(\boldsymbol{x_i}, \boldsymbol{x_j})\boldsymbol{u}$ (or $\|\boldsymbol{w}\|^2$ for the LDB case) is quadratic thus convex, then the whole objective function will be strictly convex as long as the loss function is strictly convex. Similarly, the whole objective function will be locally peseudoconvex as long as the loss function is locally pseudoconvex [29]. Moreover, the objective function covers large amount of area in the valley domain as σ takes higher values. This feature distinguishes the correntropic loss function from other loss function, and thus distinguishes the SVM models incorporating this function.

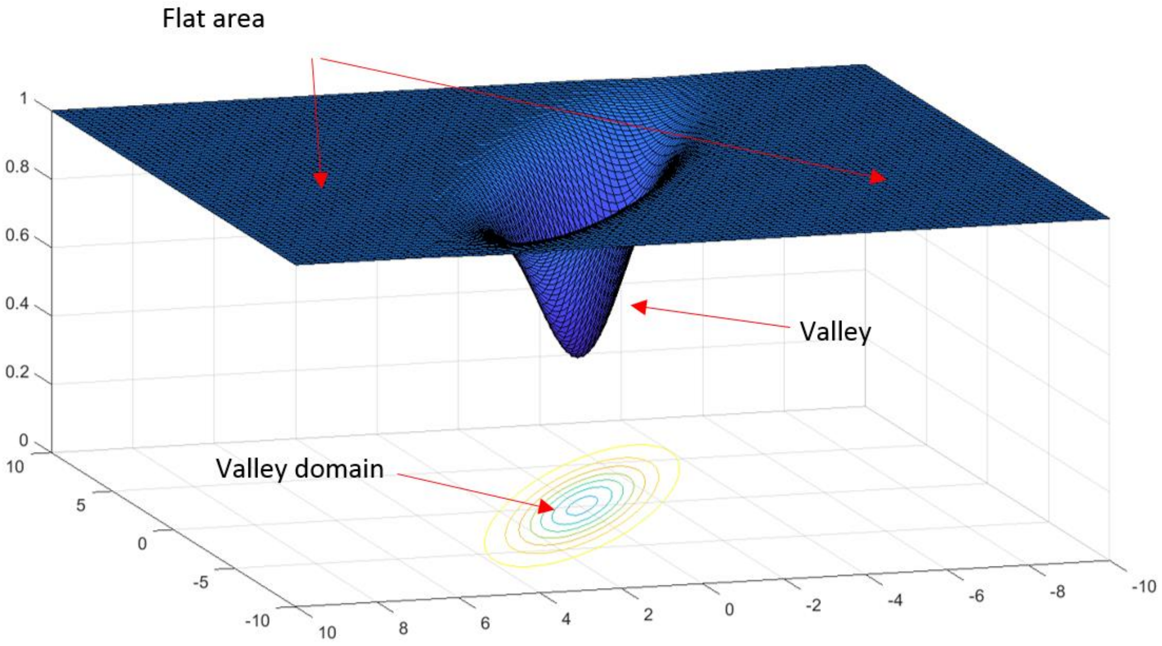
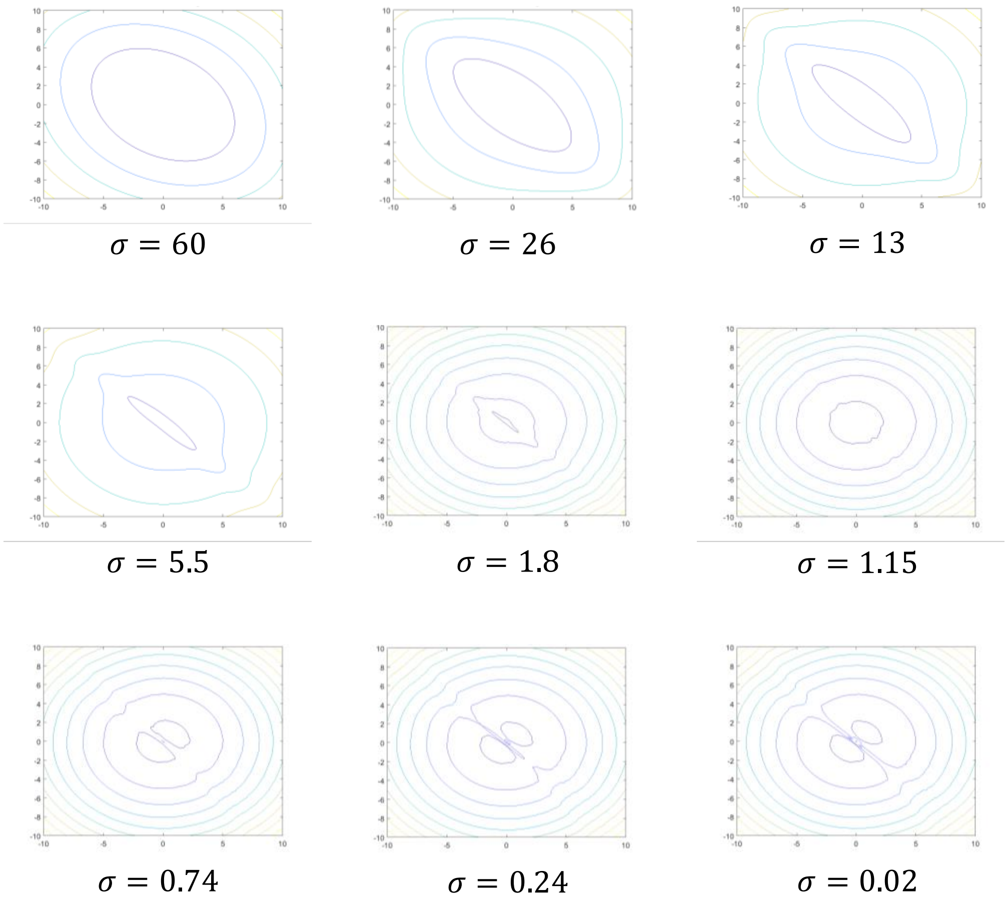


Figure 17: A plot of the C-SVM compact objective function using synthetic data. Critical areas of the graph are also identified.

It would be interesting to visualize the model behavior as the value of σ decrease. This can be done by plotting the contours of the objective function (the compact form of the model Equation (33)) and fixing the values of λ and b while varying σ. Figure (18) shows the C-SVM LDB objective function contours for a set of different values of σ, for n synthetic data points in a feature space dimension of $m = 2$. The figure also shows that the function gradually changes from being convex at very large σ, to pseudoconvex at smaller values, until it becomes non-convex with different local minima at small values of σ.

The shape of the contours and the number of local minima shown in Figure (18) are highly dependent on: the parameter λ, the structure (or distribution) of the data sets, and the existence of noise in the data sets. Figure (20) illustrates the effect of λ on the model convexity for a fixed value of σ.



$\sigma = 60$ $\sigma = 26$ $\sigma = 13$

$\sigma = 5.5$ $\sigma = 1.8$ $\sigma = 1.15$

$\sigma = 0.74$ $\sigma = 0.24$ $\sigma = 0.02$

Figure 18: Contours of the objective function ($w_1 \times w_2$) at different values of σ and for $\lambda = 10$ and the bias term $b = 1$.

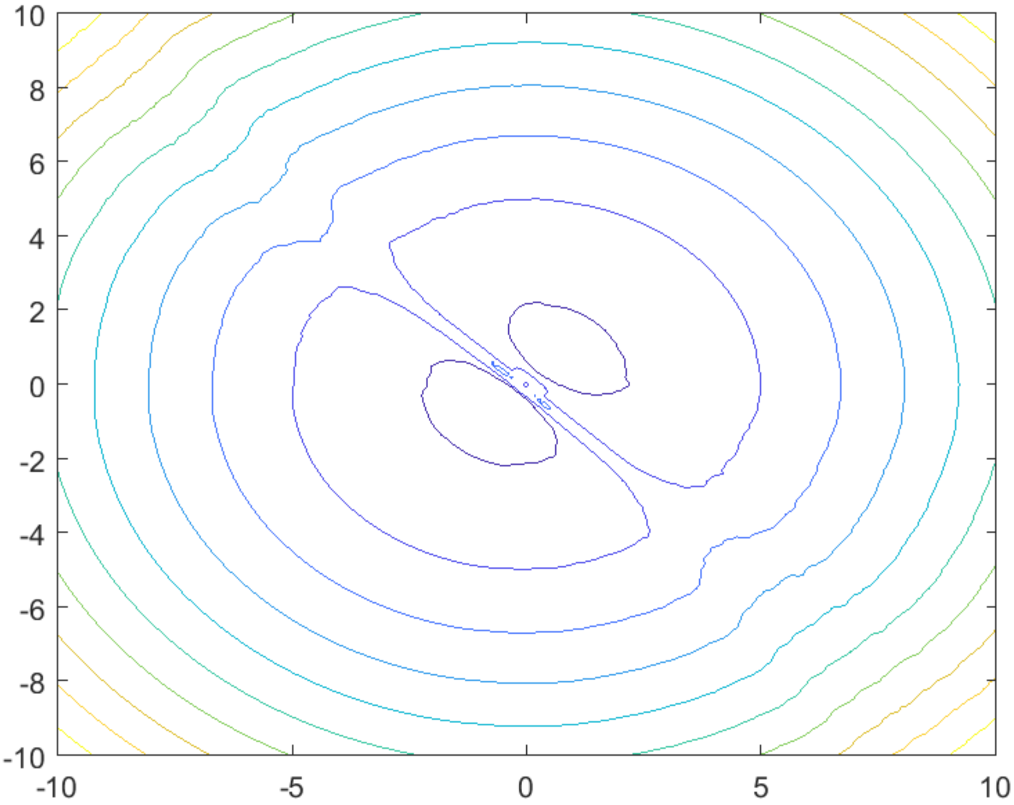
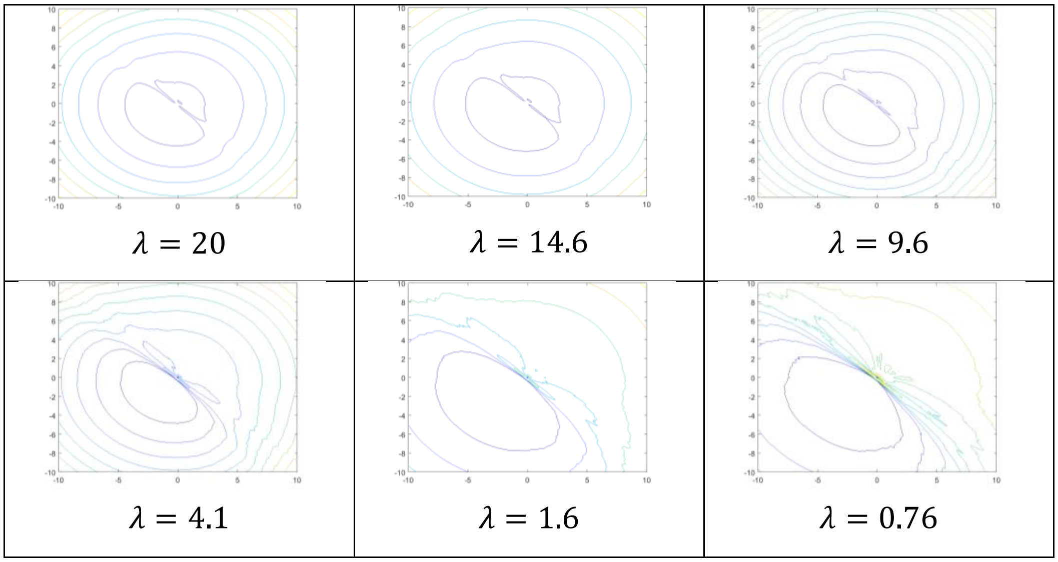


Figure 19: Contours of the objective function (w_1×w_2) for σ=0.001 and λ=10 and the bias term b=1.. Multiple minima can be seen from the plot.



$\lambda = 20$

$\lambda = 14.6$

$\lambda = 9.6$

$\lambda = 4.1$

$\lambda = 1.6$

$\lambda = 0.76$

Figure 20: Contours of the objective function ($w_1 \times w_2$) for different values of λ and for given $\sigma = 0.001$ and the bias term $b = 1$.

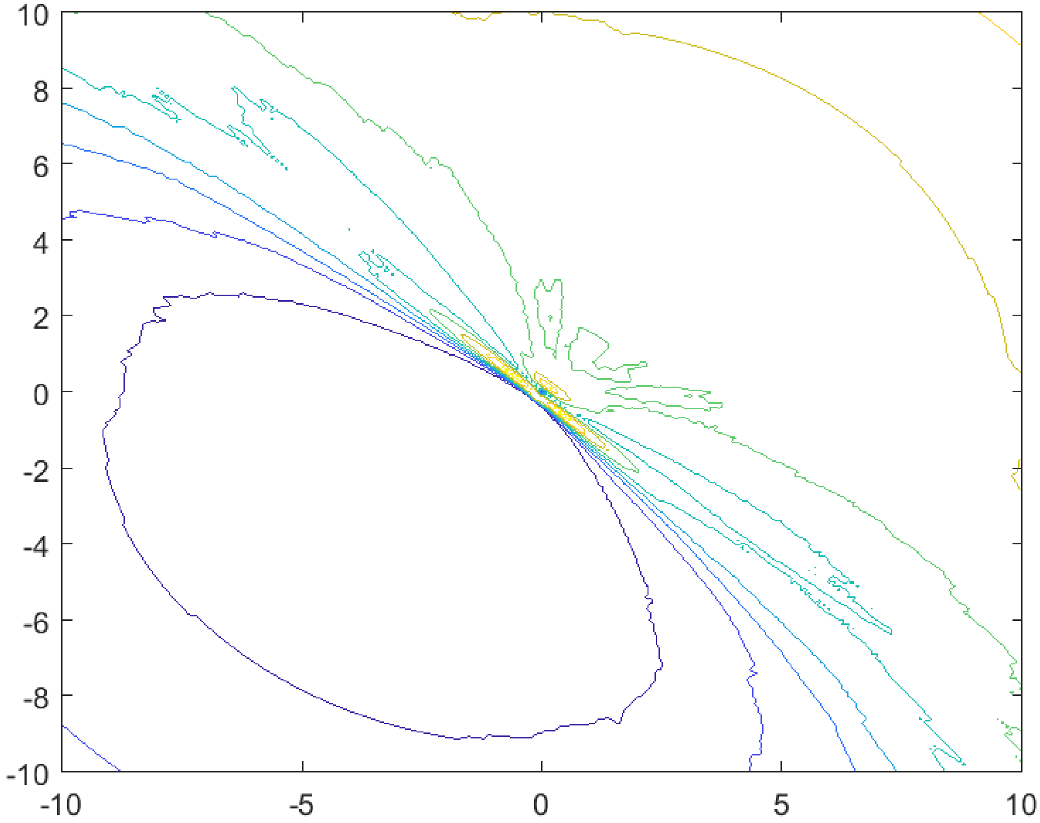


Figure 21: Contours of the objective function (w_1×w_2) for λ=0.556 and for given σ=0.001 and bias term b=1. Multiple minima can be seen from the plot.

4.2 Iterative Solution Method

The three proposed models have parameters that need to be tuned to get the optimum decision boundary. In the case of LDB models, the parameters are λ and σ only. In NLDB models, a third parameter γ is added, which is the parameter for the kernel function. As it is the case in almost all robust SVM models, there is no direct method to predetermine these parameters.

Solving the NLP problem of any of the 3 proposed models for a given combination of $\lambda, \gamma,$ and σ may not lead to an optimum solution. The domain of the objective function can be classified into two types of areas: flat areas, and the valley domain areas, see Figure (17). For low values of sigma, the flat area gets very large and the valley domain (where multiple local minima exist) becomes small. Additionally, if the initial solution (starting point) is under the flat area, then the algorithm will terminate. The mathematical explanation for terminating the algorithm in that case is the inability to get a direction to minimize the function from that point (gradient $\nabla f(\boldsymbol{u}, b) = 0$). Therefore, it is very important that the selected initial solution is located in the valley area.

The iterative solution method proposed in this thesis is a modification to the normal grid search, but specific for the proposed robust models which use the correntropic loss function. The key modification in the method is that the problem solution from the previous search is used as an input in the following search. This method utilizes the fact that the objective function is convex when the parameter σ is sufficiently large. The search starts with the highest value of σ, say σ_s, for certain values of λ and γ and with an initial solution sol_0 to solve the problem and get the new solution sol_{new}. The next iteration will be for the same λ and γ but with a slightly smaller σ such that $\sigma_{new} = \tau\sigma_s$, where τ is a fraction close to 1 (i.e. $\tau \sim 1^-$). Additionally, the initial solution for the new iteration is the final solution from the previous iteration (i.e. $sol_{0,t=2} = sol_{new,t=1}$). The above process allows the final solutions throughout all iterations to lie within the valley domain, when σ is gradually reduced throughout the iterations. As σ decreases, the objective function moves from pseudoconvexity to non-convexity, with more than one minimum point. The reducing factor τ can be set such that the change in the function shape will be slow enough for the algorithm to catch one of the newly formed local minima, if not the global minimum.

4.3 Handling Parameter σ

In this section, the proposed solution methodology is depicted on a two-dimensional data. For the purpose of illustration, a synthetic data set was developed with two classes (class A in red color and class B in blue color). The data set was designed for an LDB case, so it can be easily interpreted. The classification problem was modeled with C-SVM. As shown on Figure (22), the data set contains some outliers at the opposite side of each class. The two small groups of data in the middle may look as outliers, but they are not. That's because it is possible to come up with an LDB that would accommodate the small group in the middle and the large one of the same label in the same class. Additionally, from the definition of loss functions, noise around the decision boundary receives less attention than the noise in other locations. This is because the loss function returns low penalty cost for small marginal errors. As a result of the above discussion, the decision boundary is expected to be similar to the one shown in Figure (23).

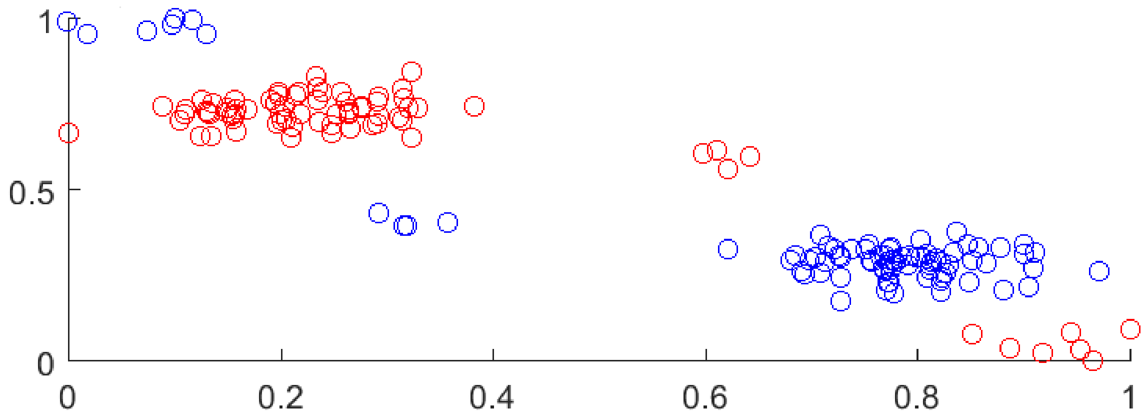


Figure 22: Two dimensional synthetic data points for two classes. Noise is represented by the blue points at the extreme top left corner and the red points at the extreme bottom right corner.

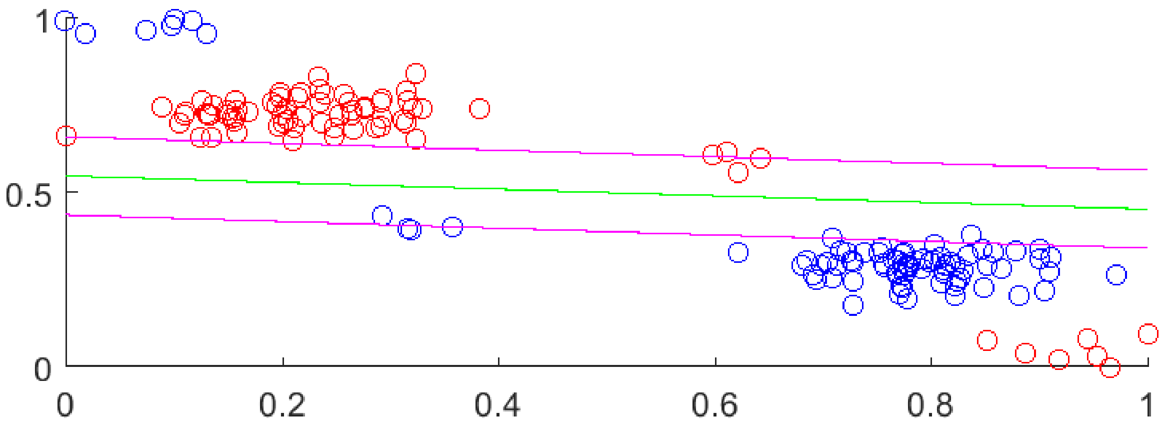
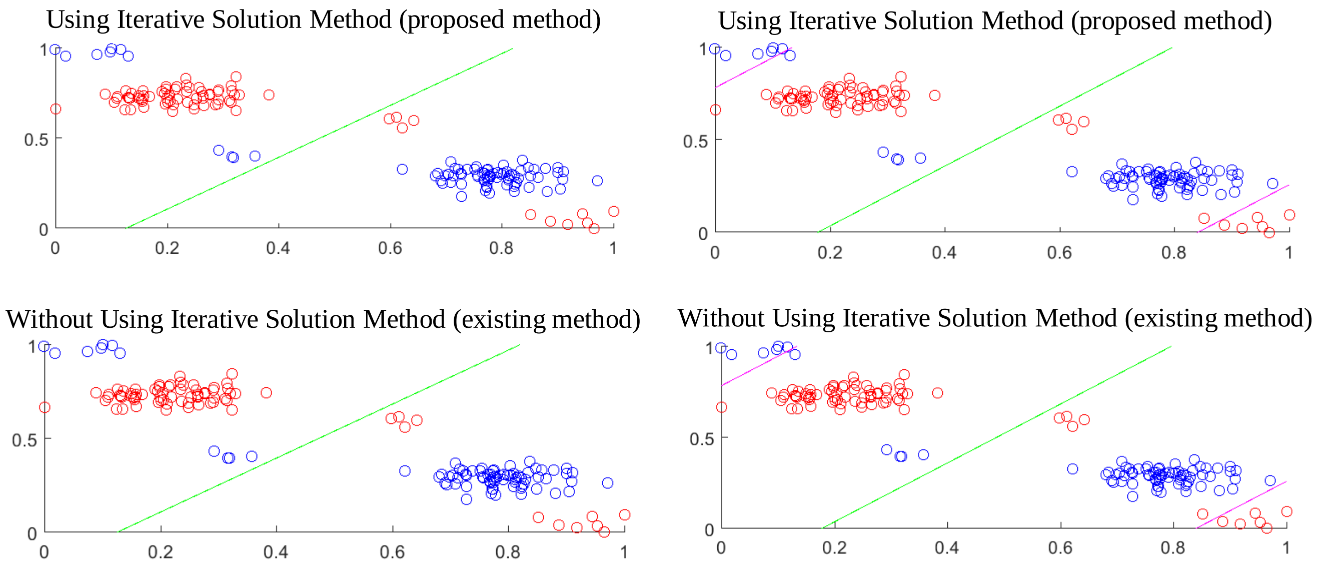


Figure 23: The predicted LDB obtained by a robust SVM model

Figures (24)-(27) show the response of the decision boundary by gradually decreasing values of σ. In Figure (24), the graphs which use the proposed iterative solution method (we will call it “proposed method” in this section for simplicity) are identical to the ones which did not use that method (we will call it “existing method” in this section). This is because σ is sufficiently high which means the optimization model is in the convex zone. This implies that any random initial solution will lead to the global minimum. With 10% reduction in σ, different decision boundaries begin to appear, as shown in Figure (25). This is because the optimization model now transforms into a non-convex model.

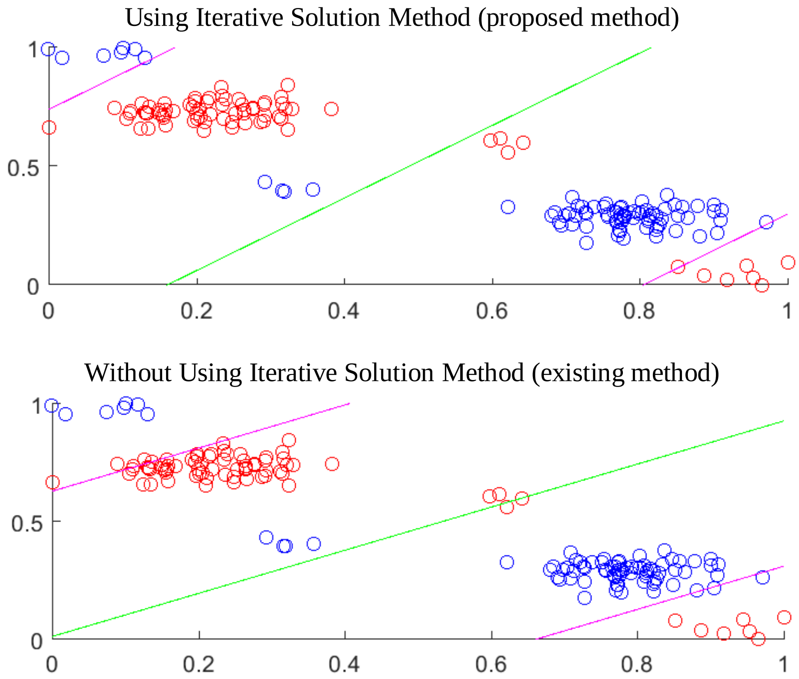
After a few more iterations, the model that is solved by the proposed method develops a decision boundary as shown in Figure (26), which is similar to the one predicted initially in Figure (23). However, the LDB obtained using the existing method seems to be bouncing between two positions, which are even far away from the optimum one shown in Figure (23). Moreover, the ultimate decision boundary developed by the proposed method, has never been obtained by the existing method throughout the experiment.



(1): $\sigma = 9$

(2): $\sigma = 3.4868$

Figure 24: The C-SVM model at sufficiently high values of σ produces the same LDB

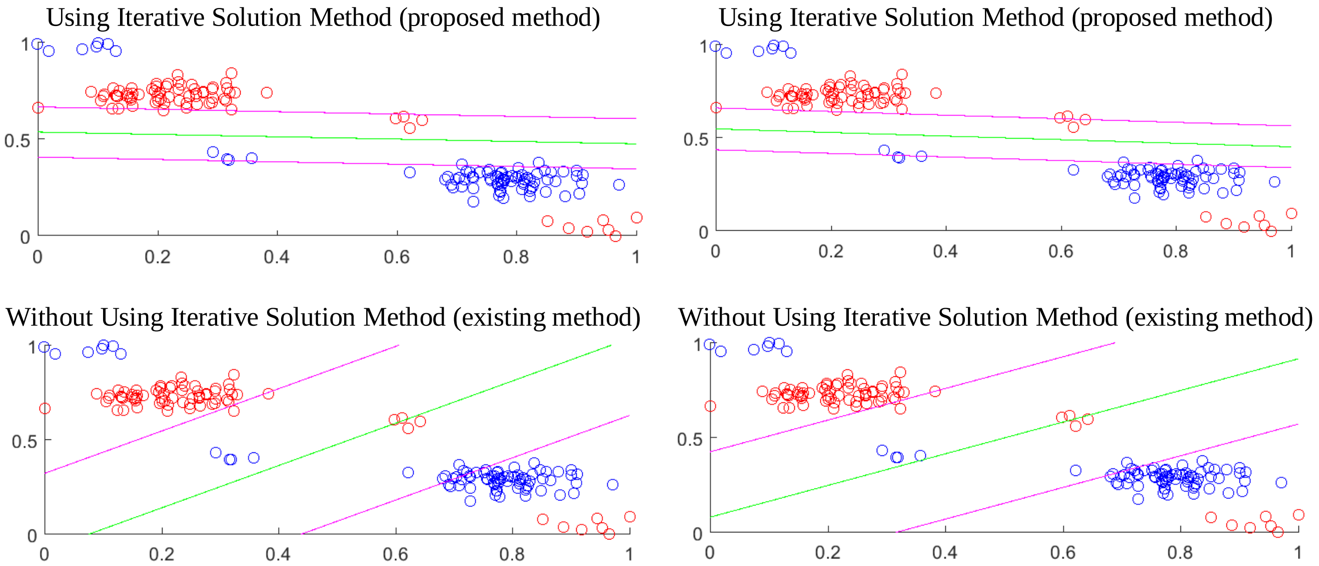


Using Iterative Solution Method (proposed method)

Without Using Iterative Solution Method (existing method)

$\sigma = 3.12$

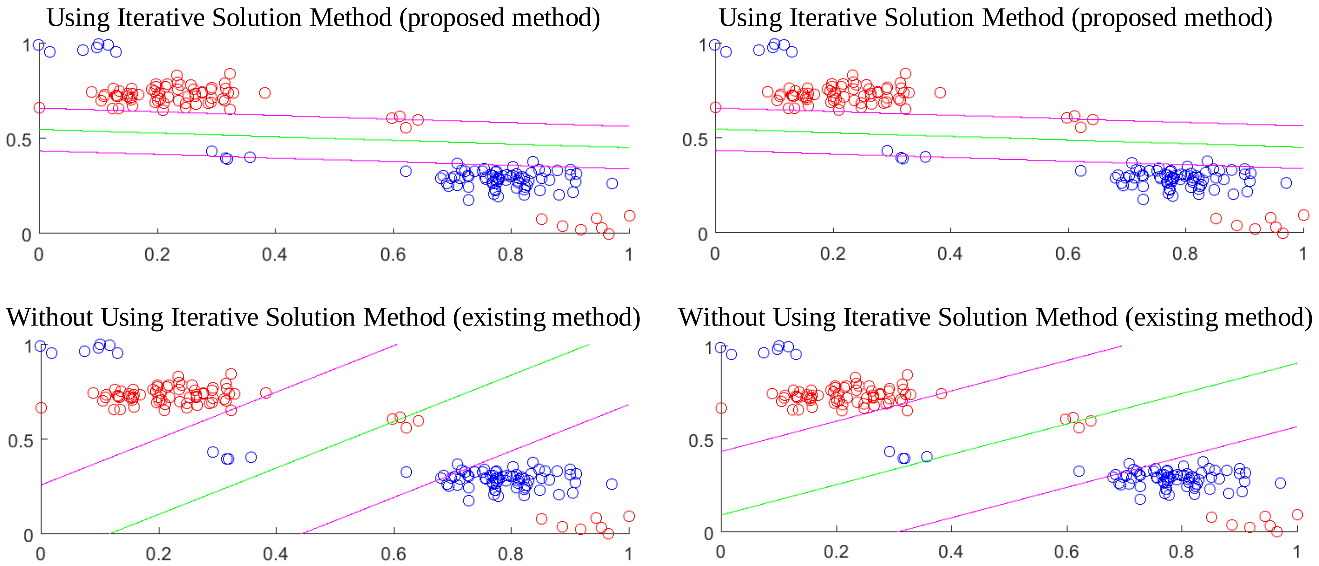
Figure 25: The LDB produced by two different solution methodologies for the same parameters. This the onset of changing from pseudo-convex function to non-convex.


Using Iterative Solution Method (proposed method)

Without Using Iterative Solution Method (existing method)

(1): $\sigma = 0.28$

(2): $\sigma = 0.046$


Using Iterative Solution Method (proposed method)

Without Using Iterative Solution Method (existing method)

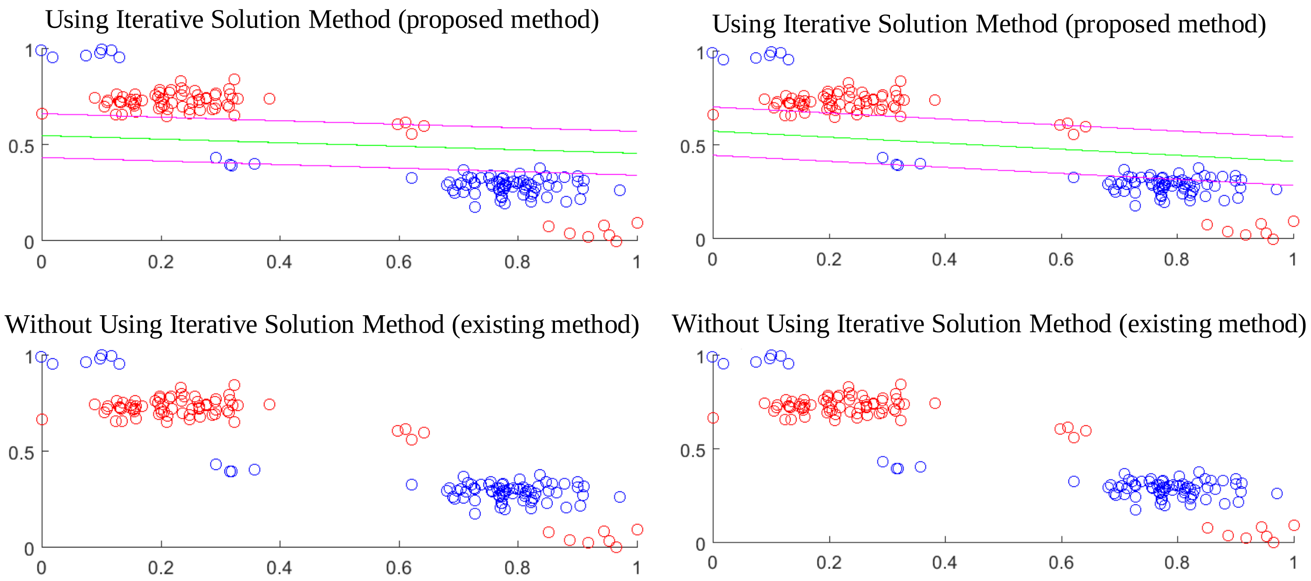
(3): $\sigma = 0.042$

(4): $\sigma = 0.038$

Figure 26: While reducing σ, the LDB obtained by the proposed method is optimal and stable, whereas the LDB obtained by the existing method is bouncing and never reached the optimal orientation.

Furthermore, at very small values of σ, the decision boundary obtained by the proposed method appear to be in the same position even when σ is as low as $\sim 10^{-7}$. This shows that the iterative solution procedure is successful in retaining the solutions inside the valley domain. On the other hand, the existing method produces decision boundaries that don’t even appear within the frame of the data sets. This can be clearly seen in Figure (27). One reason could be due to the fact that the random initial solution is unlikely to be located in the valley domain. This shows the disadvantage of directly solving the robust model for a specific value of σ since some possible good solutions may be skipped.

The parameter τ determines the smoothness of the convergence process. From our experience, it is recommended to have τ at least between 0.9 and 0.95. The value of τ can be selected to be higher or lower than the recommended value depending on different factors such as the data sets structure and the required accuracy. Furthermore, higher values of τ increases the solution time, thus the trade-off between the solution time and the smoothness of convergence needs to be studied carefully.



(1): $\sigma = 0.025$

(2): $\sigma = 1.12 \times 10^{-6}$

Figure 27: The LDB obtained by both methods and very low values of σ

4.4 Selection of Parameters ($\boldsymbol{\lambda}$, $\boldsymbol{\gamma}$, and $\boldsymbol{\sigma}$)

In this section, a grid-search based method is proposed for identifying the best values of λ, γ, and σ. The grid search process covers the one parameter λ and σ for the LDB models and the 3 parameters λ, γ, and σ for the NLDB models. We will tackle only the latter case since it is more generalized and more commonly used in the literature.

4.4.1 Grid Search for $\boldsymbol{\gamma}$ and $\boldsymbol{\lambda}$

During the training phase of the model, the gird search is conducted to identify γ^* and λ^* among certain sets of values $\bar{\gamma}$ and $\bar{\lambda}$. Each parameter can take a value in the range $[0, \infty)$, but it is not possible to conduct the search on an infinite set. This imposes the need to have a finite set of possible values, in order to have a practical search procedure. Moreover, the finite search set should have good range of values to capture the optimal value of the parameter. For example, suppose the actual λ^* is 20 but the search set is $\bar{\lambda}$ ={0.1, 0.2, 0.3, 0.4, 0.5}. In that case, the model will probably select $\lambda^* = 0.5$ in every time the search is conducted, which may not produce a sufficiently good result. A better definition of the set will be something like $\bar{\lambda}$ ={1, 5, 10, 15, 20}. The challenging part, in selecting the finite search set, is the difficulty to estimate the parameters ranges before running the experiment.

Before investigating any method to establish the search sets, two main points about the proposed models in this thesis should be considered. The first point is the scaling of the data. This indicates that for the same model but different data sets, the search sets should not differ much. Scaling plays a significant role in the learning process and all the data used in this thesis are scaled, as explained in Section 5.2. The second point is related to the objective function. The penalty term (loss function) in the objective function is exponential, hence it is bounded within the range [0, 1]. This simplifies the generation of $\bar{\lambda}$ and helps determining the boundaries of the set (smallest and largest value).

In this thesis, the search sets $\bar{\gamma}$ and $\bar{\lambda}$ are developed by trial and error. A starting search set is first created, and the model is solved several times with different training, validation (tuning), and testing sets. If the optimum parameters value in most of the runs takes the value of either boundary of the search set, that value is pushed out so larger positive (or larger negative) values enter the search set. If the optimum parameter value alternates the two boundary values of the search set, this means the set range is too small and it should be expanded to accommodate more values. An ideal search set will have all the optimum parameter values far away from the boundary elements. However, there should be some hard limits for the search sets in case the optimum parameter value is continuously taking a boundary value. In some cases, it is possible that the optimum parameter value is zero or infinity. In that case, a relatively small or large number should be established as a boundary of the search set.

The larger the size of the search set, the more accurate the learning process becomes. However, this may cause the solution time to significantly increase, due to the large number of evaluations in the process. Therefore, a good compromise between the solution speed and the size of the search set is highly required.

4.4.2 Selecting σ

Once γ^* and λ^* are identified, the model is trained over the entire training data for multiple values of σ, using the iterative solution method proposed in section 4.2. Unlike the other two parameters, the optimum value of σ is determined during this phase of training. The value of σ^* corresponds to the value of σ that gives highest training accuracy over the entire training data at $\gamma = \gamma^*$ and $\lambda = \lambda^*$. Once σ^* is identified, the decision boundary can be constructed. To sum, the optimal triplet $(\gamma^*, \lambda^*, \sigma^*)$ obtained in the training phase will be used for obtaining $(\boldsymbol{w}, b)$ for LDB, and $(\boldsymbol{u}, b)$ for NLDB.

4.4.3 Grid Search Algorithm Using Iterative Solution Method

Finding the optimum values for the two parameters is the core purpose of the learning process (known as Cross Validation), discussed in detail in the following chapter. In our proposed methodology, the iterative solution concept and the effect of σ are integrated in the grid search of multiple parameters. This forms a sound procedure for optimizing the parameters. Algorithm (1) shows the proposed grid search method:

Algorithm 1: Proposed Iterative Solution Algorithm

```
Set $\gamma = \gamma_s, \lambda = \lambda_s,\ \sigma = \sigma_s$
Set the reduction factor $\tau$ for $\sigma$
While ($\gamma \neq \gamma_e$) {
  Set $\lambda = \lambda_s$
    While ($\lambda \neq \lambda_e$) {
      Set $\lambda = \lambda_s$,
      Set $\boldsymbol{u}_{init}$ and $b_{init}$ randomly
        While ($\sigma \geq \sigma_e$) {
          $\boldsymbol{u} = \boldsymbol{u}_{init}$ and $b = b_{init}$
          Solve NLP model
          $\boldsymbol{u}_{init} = \boldsymbol{u}^*,\quad b_{init} = b^*$
          $\sigma = \tau\sigma$
          }end
      Update $\lambda$
    }end
Update $\lambda$
}end
```

After solving the NLP model for all values of γ, λ, and σ, the validation accuracy for every parameter combination is obtained. The validation accuracies are bundled in a 3D matrix, where each dimension corresponds to one of the three parameters. The optimum parameters ($\gamma^*$and λ^*) are the ones corresponding to the maximum validation accuracy in the 3D matrix. The model is then solved using these two optimum parameters to find σ^* and the decision variables $\boldsymbol{u}$**,** $\boldsymbol{w}$**,** and b, depending whether the problem is for LDB or NLDB.

4.5 Solving Method for CMD-SVM Model

The solution for the CMD-SVM model follow the same concept and methodology explained in the previous sections. However, the structure of the CMD-SVM model is different from the other two proposed models since membership degrees are used instead of the loss function. The model requires a different solution approach to handle membership degree α in the objective function. Therefore, solving the model, which is step number 10 in Algorithm (1), is replaced by Algorithm (2) for CMD-SVM model.

We are proposing an iterative solution method, see Algorithm (2), where α is updated in each iteration for a specific combination of γ, λ, and σ. The algorithm starts with assigning a random initial solution $\boldsymbol{u}_0$ and b_0. The initial solution is then used to find α_0, which is substituted in the objective function. The objective function is solved for $\boldsymbol{u}_1$ and b_1 and the process is repeated until no significant improvement is observed in the objective function value.

Algorithm 2: solving the CMD-SVM at given γ, λ, and σ

```
Set $t = 0$, $\boldsymbol{u}_0$, $b_0$= random, $\varepsilon = 0^+$, $f^t = 0$, $f^{t-1}$= M
While ( $\frac{|f^t - f^{t-1}|}{f^{t-1}} \geq \varepsilon$ ) {
    Calculate $\alpha_i^{(t+1)}$ $\forall\, i$ using $\boldsymbol{w}_t$ and $b_t$
    $t = t + 1$
    Solve the NLP for $\boldsymbol{w}_t$ and $b_t$
} end
```

The parameter α is constant when the problem is solved for $\boldsymbol{u}$ and b, which makes the problem convex at that particular iteration t. Because the objective function is convex, there are plenty of fast and efficient algorithms for solving the convex problem. Despite the admired convexity of the objective function, it is very difficult to mathematically prove that the solution will always converge. Alternatively, a simple experiment was conducted to test the solution convergence of the model empirically. For a certain γ, λ, and starting from large value of σ, the model was solved per Algorithm (1). The algorithm was executed multiple times with different initial solutions. It was found that the solution produced by the CMD-SVM model is empirically converging. One drawback of using CMD-SVM is the increase in the solution time, which is due to a new loop added to the overall learning process.

CHAPTER 5

NUMERICAL EXPERIMENTS

Numerical experiments are crucial in any research since they demonstrate whether the proposed work meets the expected deliverables. Furthermore, they allow comparing the quality of the proposed work to existing works in the literature. In this chapter, we test the performance of the three proposed models and compare it to the original SVM and PSVM, and existing robust methods. The main objective of this chapter is to show that the proposed models are robust when outliers exist in the data sets.

This chapter starts with explaining a key concept known as Cross Validation (CV), which is a milestone in solving any data analysis problem. Next, the solution algorithm is discussed followed by the experiment setup. After that, LDB classification problem using synthetic data set is illustrated, followed by the well-known Double Banana (crescent) data set for the NLDB case. These two data sets are in 2D (feature space dimension $m = 2$), which means the data sets and the decision boundary can be visualized. Lastly, the performance of the proposed NLDB algorithm is illustrated on real life data sets, obtained from public data repositories.

5.1 Cross Validation (CV)

Recall the classification process demonstrated in Figure (1). The third stage "Train the model" is where CV takes place. In fact, CV is all about estimating the model parameters. CV aims to come up with the optimum values of the model's parameters (λ, γ) by evaluating the model at different combination of those parameters. In this thesis, the combinations are obtained through the proposed grid search with the iterative solution method, discussed in the previous chapter.

Generally, the first step of the CV process is to randomly split the data set into two subsets: one subset is for training "TR" and one for testing "TS". The training subset is then further split (randomly) into two subsets (say TR1 and TR2). For a given combination of the two parameters, TR1 is used for developing the decision boundary, and TR2 is used to evaluate (validate) that decision boundary. Then the parameter combination that corresponds to the best validation performance measure is selected to build the ultimate decision boundary using TR. For example, if λ has three potential values $(\lambda_1,\ \lambda_2,\ \lambda_3)$, and γ has two potential values $(\gamma_1,\ \gamma_2)$, then we will have 6 different combinations of $(\lambda_q,\ \gamma_r)$ to evaluate. For each combination, the model will be solved using the subset TR1 to obtain the optimum decision boundary variables $\boldsymbol{u}$ and b. After that, the resultant rule function from the obtained optimum variables is used to classify the data instances in subset TR2, and calculate the classification accuracy of the model w.r.t TR2. This accuracy is called "cross validation accuracy" or simply "validation accuracy". There are several accuracy measures used in classification. Different accuracy measures provide different interpretation and insight. Some of the measures are applicable in certain conditions, such as balanced or imbalanced data [30]. Depending on the type of CV, the validation accuracy is obtained K times for each parameter combination, where each time TR1 and TR2 are regenerated randomly. The parameter combination corresponding to the best average accuracy is selected for further analysis. The last training step is to train the model with the entire training subset (TR1+TR2) using the optimum parameters ($\lambda^*,\ \gamma^*$) to obtain σ^* and the optimum $\boldsymbol{u}$ and b, hence the rule function.

The last stage in CV is testing, which aims to evaluate the rule function produced in the training phase. The TS is not involved in any way in the training process described earlier. It can be considered as a set of new instances from the future, which need to be classified based on the obtained rule function. The performance measures obtained by the testing subset, such as accuracy and sensitivity, are called “prediction accuracy” or “prediction sensitivity”, respectively.

The most important performance measure is the accuracy. Different measures (formulas) were developed to calculate the prediction accuracy of a decision rule. Each formula has its own meaning and interpretation of accuracy, which may suit certain classification problems or areas of applications. Moreover, some accuracy measures are preferred for certain data configuration, such as the case of class imbalanced data sets (when the two classes are extremely different in size). For example, in case of class balanced data, normal accuracy can be used which is formulated as follows:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{60}$$

In case of class imbalanced data, balanced accuracy measure is sounder and more realistic, which is formulated as:

$$BACC = \left(\frac{TP}{TP + FP} + \frac{TN}{TN + FN}\right)/2 \tag{61}$$

It is convenient to have the testing results summarized in a tabulated format. The testing results are mainly the counts of the data points that are correctly/incorrectly classified as positive; and the points which are correctly/incorrectly classified as negative. This test summary table is known as the “confusion matrix”. Table (2) shows the configuration of the confusion matrix, which only provides the counts of TP, FN, FP, and TN data points. These four numbers are the building blocks for most of the performance measures. The confusion matrix minimizes the time and effort of computing the accuracy especially if multiple types of accuracy measures are to be computed. Additionally, it represents the results in neat and organized manner, which usually minimizes the risk of making a mistake in the calculations.

Table 2: Typical confusion matrix

	Classified as Positive	**Classified as Negative**
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

There are several methods of CV and each method has a different training procedure and data subset configuration. The three most common methods are:

Hold-Out Method: This is probably the simplest method that avoids overfitting [3]. In this method, the data is first split into two groups: the training subset and the validation subset. The validation subset is held-out for validation in a later stage, hence the name. The training subset is used to train the model and obtain the optimum model parameters. The validation subset is then used to evaluate the model performance by classifying those validation instances; and to compare the results with the true labels, using the performance measures discussed previously.

Leave-One-Out Method: In this method, the model is solved n times (total number of data points). In each iteration, one point is left for validation and the remaining $n-1$ points are used for training (solving the model). This method is preferred when n is small, where other methods become impractical [31].

K-fold method: This method is a compromise of the previous two methods and it is the most commonly used one in the literature. The training subset is divided evenly into K number of folds and the model is solved K times. In each iteration, one fold represents the CV testing subset and the remaining $K-1$ folds are assigned for training. Normally the number of folds K takes a value between 1 and 10.

Insufficient randomness in selecting the training and validation subsets in the Hold-Out Method may introduce bias to the CV process. For example, all outliers may get assigned to the validation set, which will lead to a very low validation accuracy measure. This bias can be avoided by repeating the whole CV process multiple times and averaging the performance measures across the repetitions. In this thesis, the hold-out CV method with repetitions is used.

5.2 Experiments Setup

This section introduces the setup of the numerical experiments conducted in this chapter. The experiments are coded in Matlab, which is then linked to GAMS using GDX files to solve the optimization problems. SNOPT solver is selected to solve the NLP model, since it is suitable for problems with large number of variables and constrains. Generally, this solver requires less matrix computation and less evaluation of the objective function compared to the other solvers, which impacts solution speeds. All the experiments are run on a laptop with Intel Core i7-3610QM CPU and an installed RAM of 6 GB. The OS is Windows 7 Professional 64-bit.

The experiments are generally setup similar to the ones in [15]. The hold-out cross validation method is used with the following data split: 60% for training, 20% for validation (tuning), and 20% for testing. The validation and testing accuracies are measured using the formula in (60). The random functions in Matlab are controlled by the “GlobalStream” facility to ensure consistency in splitting the data sets and selecting the outliers. This should eliminate any bias that may occur because of different selection of the data points, allowing a fair comparison between different models. The stream is reset before solving a classification problem for a different noise level (scenario) or before solving for a new model.

For all the experiments in this chapter, the noise is added to a dataset by flipping the class labels of some randomly selected data points. For instance, if 10% noise is to be added to the data set of $n = 100$, it means we randomly select a total of 10 points from the data and flip their class labels. The resultant data set becomes the new set to be used in the CV process. It is important to mention that the random selection of the 10 points may encounter some bias if the selection is not balanced across the two classes. This is a critical point that needs to be incorporated in the experiments, which will be explained in the coming sub-sections.

All the data sets are scaled to [0, 1] scale prior to establishing the models. Data scaling has a significant impact on the training performance [32]. It also normalizes the feature space across the data set to prevent any biased influence of a certain feature due to extremely high or low values. For example, consider a data set of 50 points with a two-dimensional feature space. Feature 1 is temperature which normally takes a value between [100, 200] and Feature 2 represents a status of *something* which takes an integer value between [1, 4]. Training the model with this data may affect the performance of the training process due to the extremely higher values of Feature 1 compared to those of Feature 2. Therefore, the data needs to be scaled to avoid this effect. Several methods are available to scale data such as the Min-Max and the Z-score methods [32]. In this thesis, the Min-Max method is used, where the values of the j^{th} feature are mapped to the new scaled values as follows:

$$x'_{i,j} = \frac{x_{i,j} - x_{min,j}}{x_{max,j} - x_{min,j}} \text{ }, \forall \text{ } i = 1,2,\cdots,n \tag{62}$$

where $x_{min,j}$ is the smallest value in the j^{th} feature across the whole dataset.

For all the experiments in this chapter, the search sets of the parameters were developed under the guidance from Sections 4.3 and 4.4. The search sets for γ and λ are developed on a logarithmic scale. The regularization parameter of the objective function has a search set $\bar{\lambda} = \{\lambda: 10^{-8+l} \mid l = 0, 1, \cdots, 11\}$. The lower bound of the set is $\lambda = 10^{-8}$ which is small enough to be considered as zero. The upper bound is $\lambda = 10^{3}$ which is sufficiently large to be considered as infinity. The non-linear overfitting parameter has a search set $\bar{\gamma} = \{\gamma: 5^{-5+l} \mid l = 0, 1, \cdots, 7\}$. The shaping parameter σ starts at $\sigma_s = 30$ and it ends $\sigma_e = 0.1$ with a reduction factor $\tau = 0.9$.

As shown in Algorithm (1), the nested loops are expected to slow down the solution time heavily. Therefore, the Grid & Multi-Threading facility in GAMS is used to perform parallel computing. Instead of solving the NLP for each γ, λ, and σ one at a time, the NLP is solved with all combinations of γ and λ in parallel at a single value of σ. This is achievable because the solution ($\boldsymbol{w}$, b) or ($\boldsymbol{u}$, b) at a certain combination (γ, λ) is independent on the other combinations. On the other hand, the solution at any σ is directly dependent on the solution at the previous σ, as per the iterative solution method explained in Section 4.2.

5.3 Synthetic Illustration Problems

It is always preferred to graphically present the data of both classes along with the decision boundary. This helps in verifying the mathematical interpretation of the results and perhaps comparing different outcomes easily. Because it is very difficult to find proper real-life data that can be plotted in 2D or 3D (i.e. feature space is $m = 2$ or $m = 3$, respectively), synthetic data is used. Two experiments were developed: The Segmented Blocks experiment for the LDB case, and the Double Banana experiment for the NLDB case.

5.3.1 Segmented Blocks Experiment for LDB

In this experiment, the data is evenly split into two halves: the right-hand side half representing once class and the left hand side half representing the other class. Each class is divided into a top half and a bottom half. This means a top or a bottom half within a class constitutes quarter the total data, and we call it a “block”. Each block is divided vertically into three segments laid out next to each other. Figure (28) describes the data layout.

In an ideal scenario with no noise involved, the decision boundary should be a straight vertical line separating the two classes. As the purpose of this experiment is to test the models’ robustness, noise will be added to the data in order to test the response of the LDB to that noise. To achieve the maximum effect on the LDB position, the outliers are placed in Block 1 and Block 4. Having the outliers in those two blocks will try and push the decision boundary to move counter-clockwise. Placing the outliers in Blocks 2 and 3 should also work the same.

As explained in a previous chapter, the reason behind noise sensitivity in SVM and PSVM is the error function. That is, higher penalties are assigned for higher marginal errors. As the outliers are close to the “division gap” (the vertical gap around where the two classes are separated) the penalty remains minimal. On the other hand, if the outliers are on the outboard of the data, then the penalty will be significantly huge, causing the decision boundary to move counter clockwise to minimize the total error. Therefore, the location of the outliers has a strong role in determining the position of the LDB.

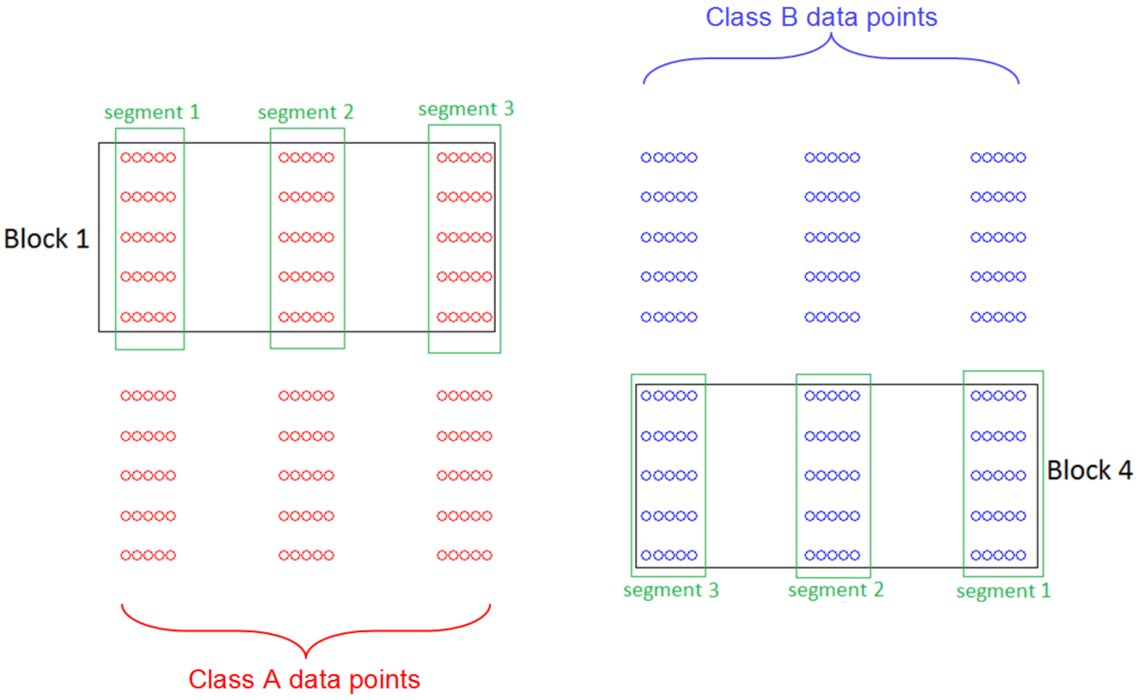


Figure 28: The structure of the data set used in the Segmented Blocks experiment

The three segments in each block are utilized to evaluate each model's performance, by adding noise at different locations. The experiment is divided into 4 scenarios. In Scenario-1, the problem is solved with no added noise. In Scenario-2, the problem is solved by adding noise to the third segment of each block. In Scenario-3, the noise is added to the second segment only of each block. The same process is replicated for the Scenario-4, but the noise is added to the last segment (Segment 1) of each block. This way, the robustness can be evaluated by observing the change in the LDB position as noise move from one segment to another, away from the division gap. The total number of data points generated for this experiment is 300. The points are split evenly into two classes, where class A is highlighted with the red circles and class B is highlighted with the blue ones.

Every model is trained to obtain the optimum λ. This is done by solving the model using the training data, and then calculating the accuracy using the validation data set. This process is repeated 20 times. In each time, the data is randomly split into training and validation sets. After that, the accuracies are averaged across the 20 repetitions, and the λ corresponding to the highest average accuracy (λ^*) is selected. The model is then re-solved with λ^* using the entire data to obtain the optimum variables $\boldsymbol{w}^*$ and b^*. Those optimum variables are used to plot the LDB.

It is expected that the noise-sensitive models will heavily respond to the addition of the outliers and will try to accommodate them into the relevant classes. However, the robust models are expected to remain unchanged, or perhaps experience some minor changes. Figures (29)-(33) show the outcome of the experiment, where the green lines represent the decision boundaries.

For a given γ and λ, the SVM and PSVM models produce one single solution ($\boldsymbol{w}$ and b), whereas the proposed robust models produce a number of solutions equal to the number of values for σ. The logical step in this case is to select the solution from the σ that returns the highest accuracy. In most cases, it is possible to have multiple solutions giving the highest level of accuracy for the same combination (γ, λ). In other words, multiple values of σ may provide different solutions but they all give the same (highest) accuracy. For the optimum combination (γ^*, λ^*), all the best accuracy solutions are incorporated in the following plots. Thus, multiple decision boundaries are shown on the plot of each of the proposed robust models, forming a “cloud” of optimum boundaries.

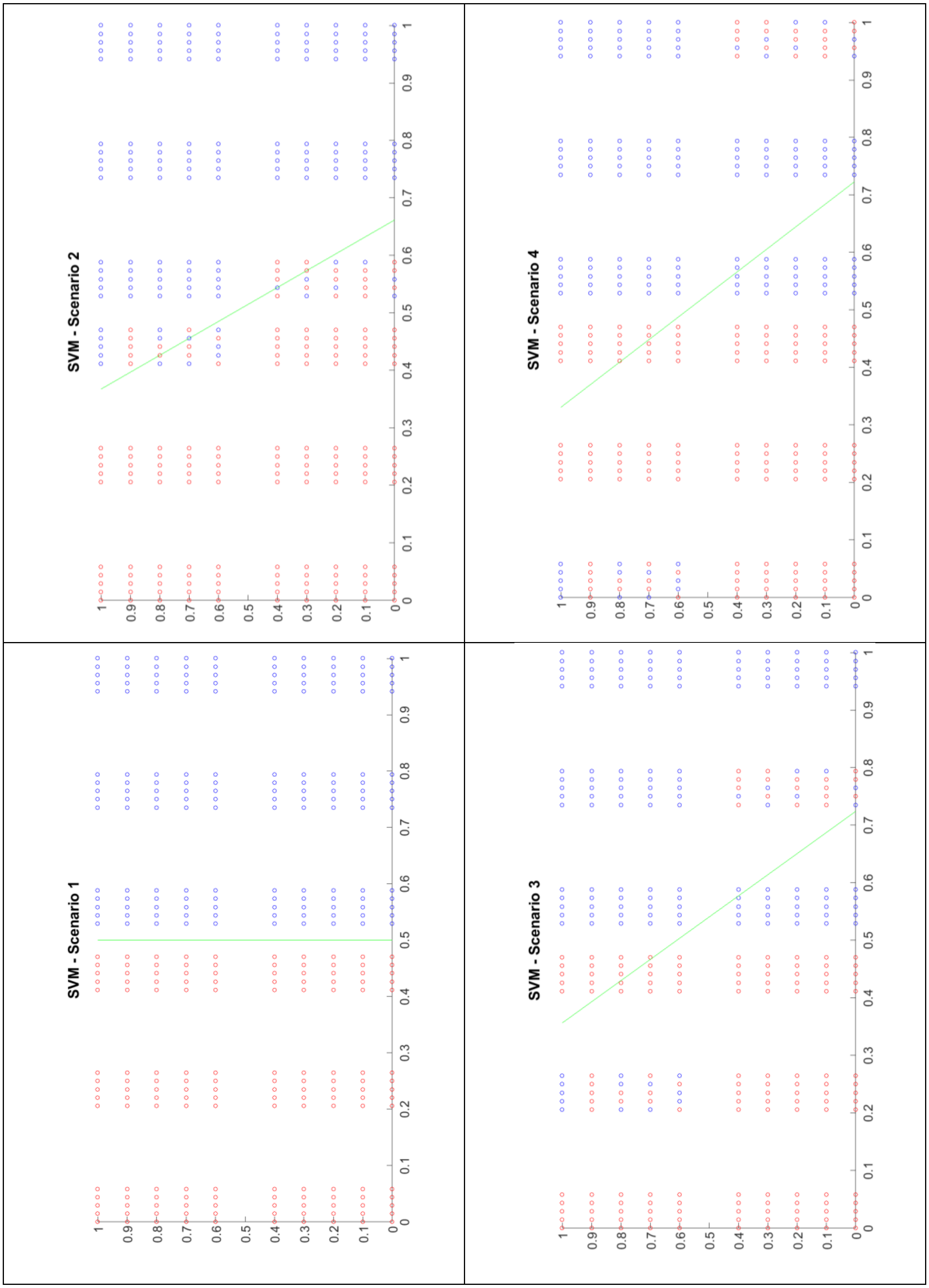


Figure 29: The LDB obtained by the SVM model for 4 different scenarios

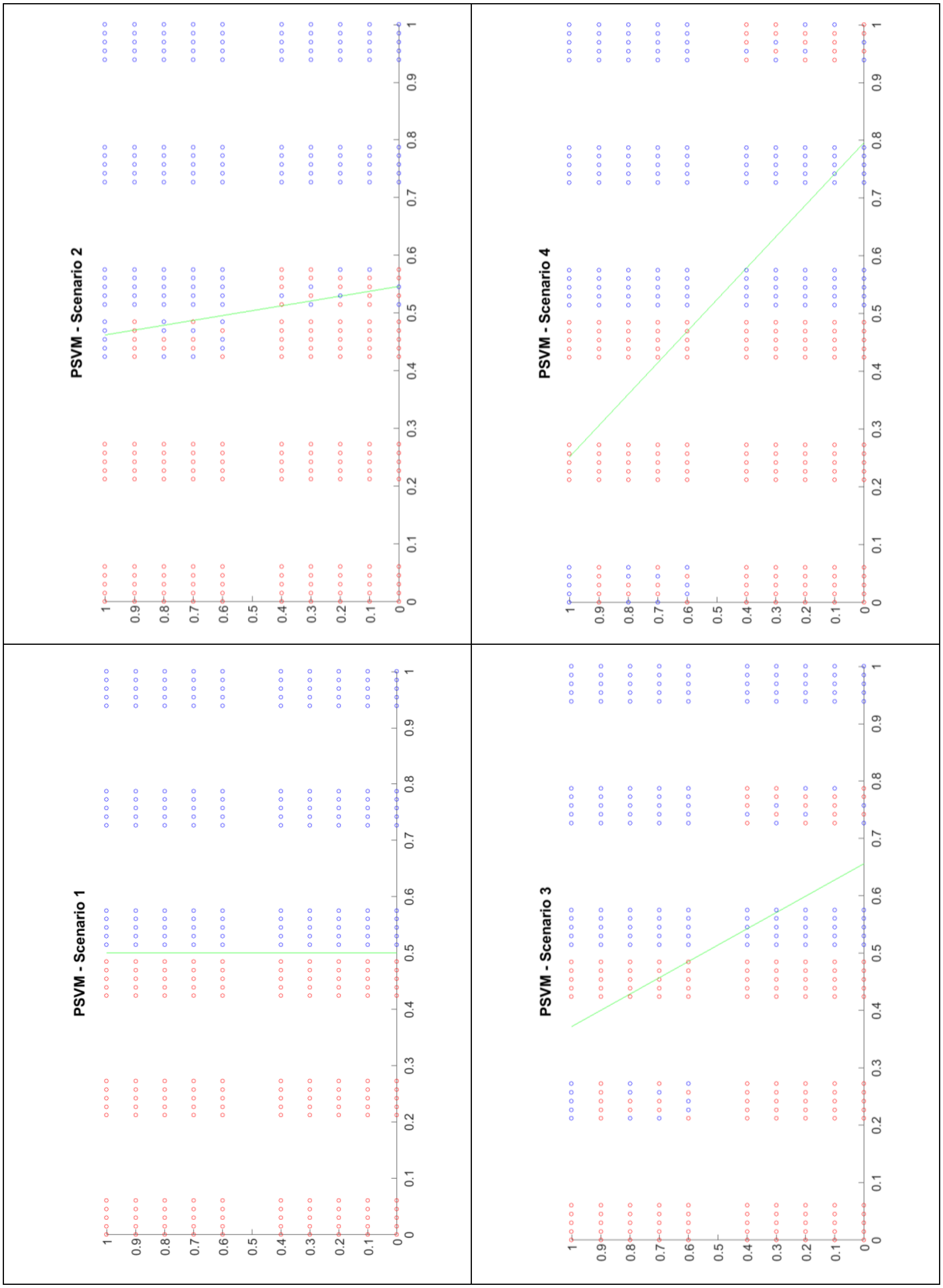


Figure 30: The LDB obtained by the PSVM model for 4 different scenarios

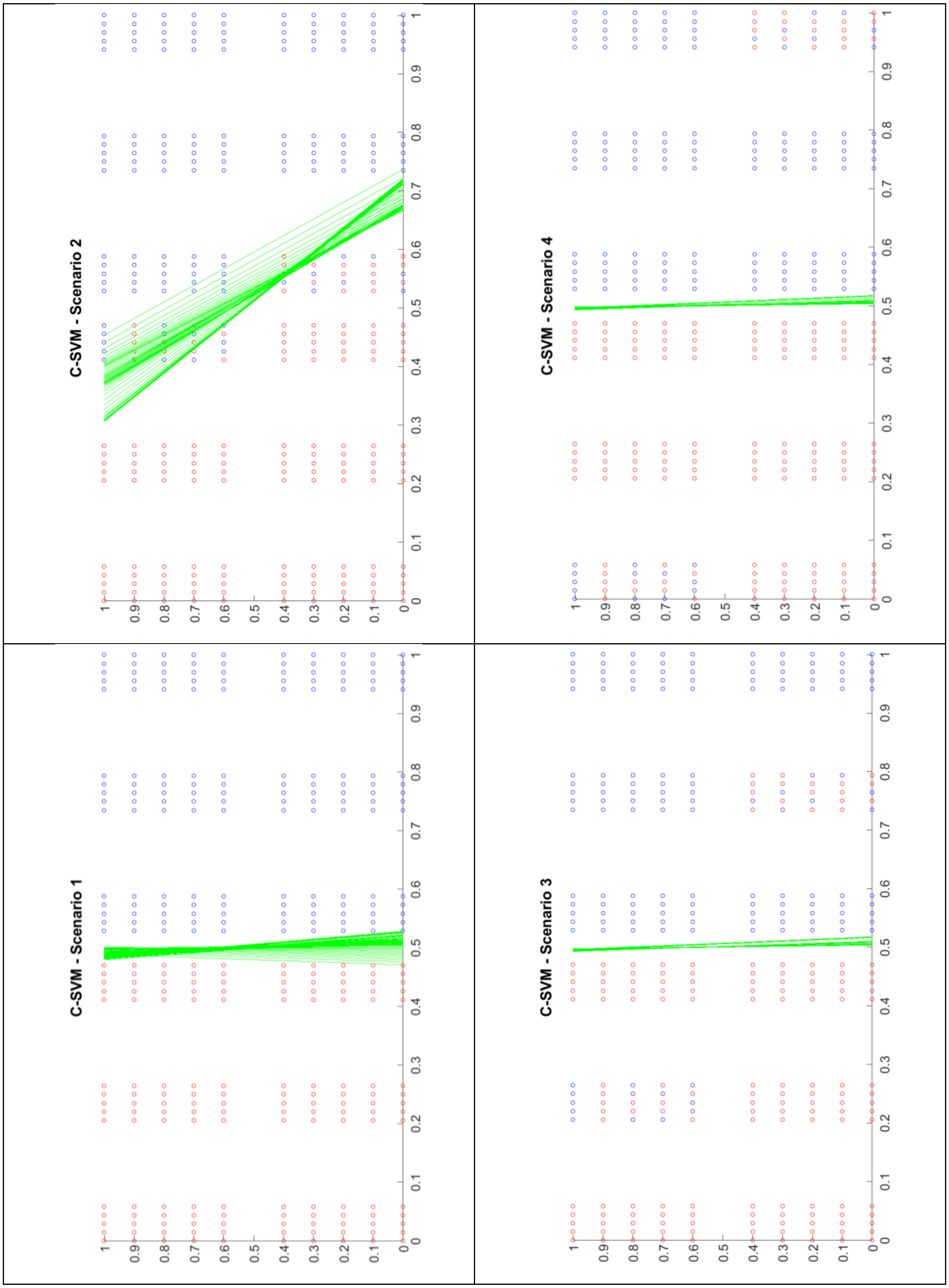


Figure 31: The LDBs obtained by the C-SVM model for 4 different scenarios

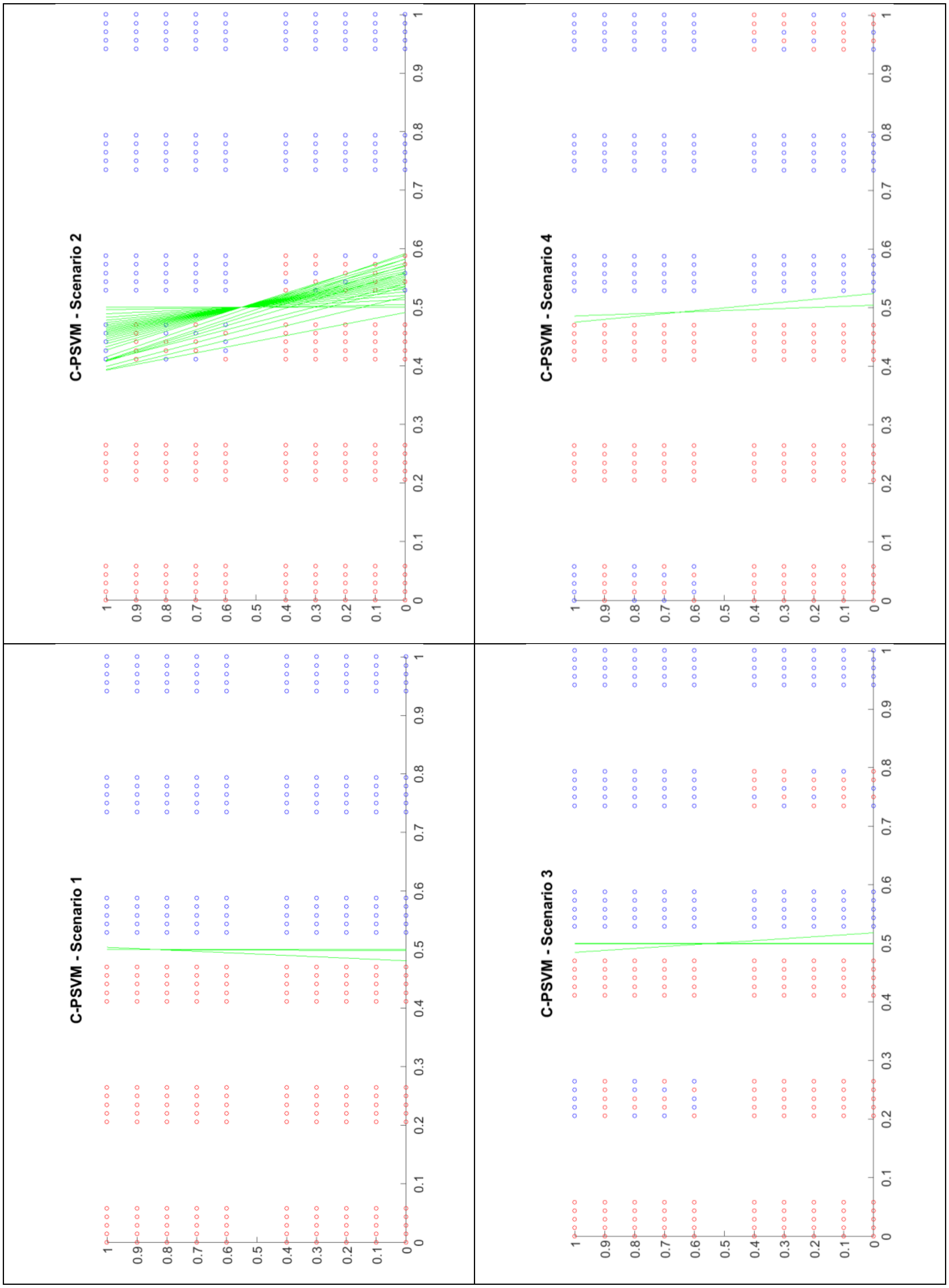


Figure 32: The LDBs obtained by the C-PSVM model for 4 different scenarios

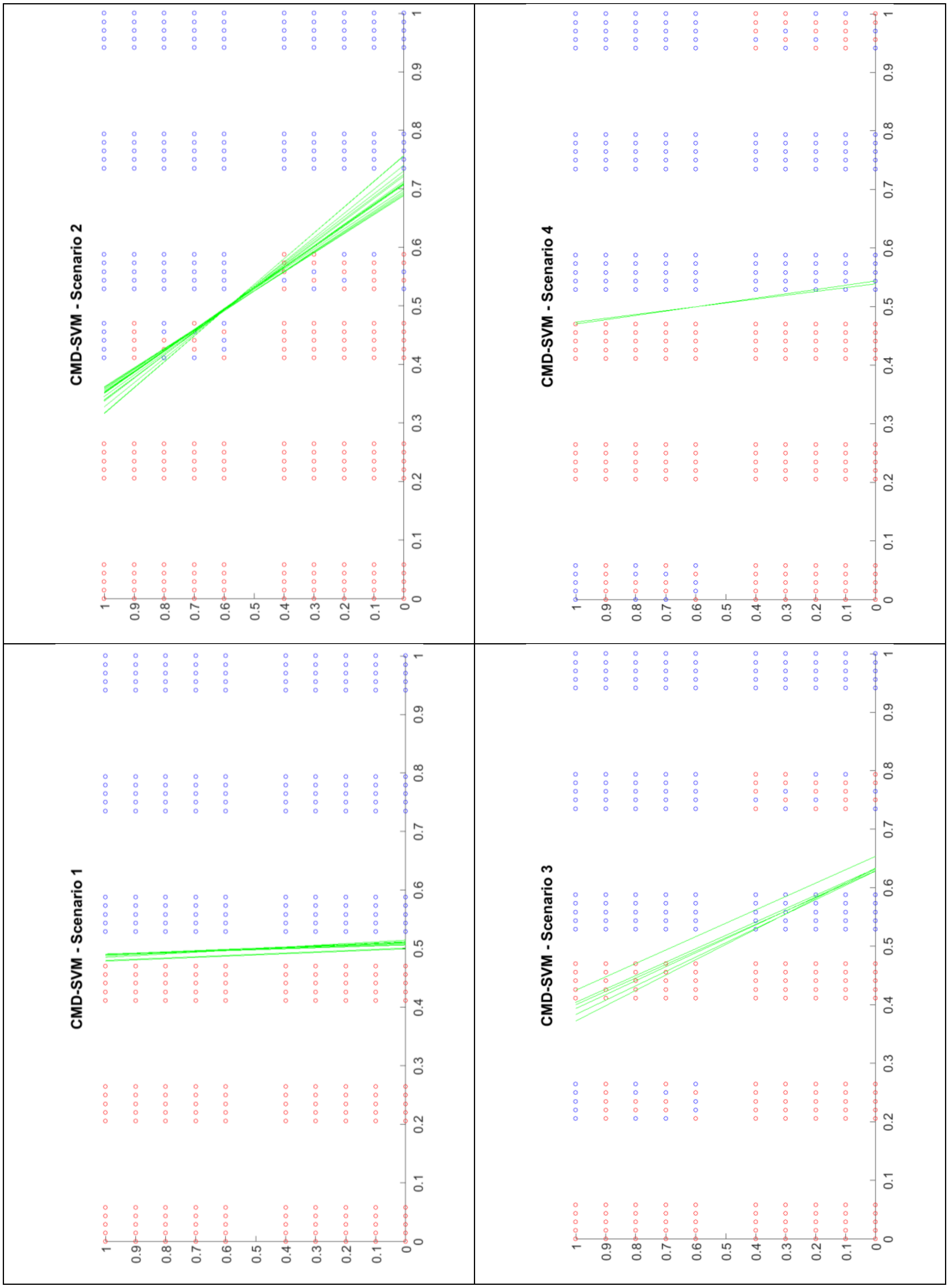


Figure 33: The LDBs obtained by the CMD-SVM model for 4 different scenarios

In Scenario-1, all models performed similarly by splitting the two classes in the middle with a vertical decision boundary. It can be seen that the LDB for the SVM and PSVM is perfectly vertical and exactly in the middle. This is expected since both models tend to maximize the margin between the two classes. In the three proposed robust models, multiple boundaries are shown, and it can be seen that there is at least one perfectly vertical LDB that is located exactly in the middle. The other boundaries are also good solutions since they provide a 100% accurate separation.

Scenario-2 is quite tricky, and it can be misleading. Because the outliers are scattered around the boundary, slanted LDB can correctly classify (separate) some of these outliers to produce a better accuracy. Note that these decision boundaries are selected because they achieve the highest accuracy against the testing samples, which are not highlighted on the plots.

In Scenario-3 and Scenario-4, the outliers are placed far from the division gap, thus started to enforce high penalties on the models. The SVM and PSVM reacted heavily to those outlier whereas the proposed robust models showed less sensitivity and produced an LDB with almost 100% accuracy. Although the CMD-SVM model performed below expectation in the 3rd scenario, it shows an acceptable solution in the 4th scenario. In general, we can confidently conclude that the three proposed models showed excellent robustness and less sensitivity to outliers compared to the conventional models. Moreover, the C-SVM and C-PSVM models outperformed the CMD-SVM model, perhaps due to the inaccuracy caused by the stopping criteria in Algorithm (2).

5.3.2 Double Banana Experiment for NLDB

The Double Banana or Double Crescent data set is a well-known set used for evaluating NLDB models. The purpose of this experiment is to assess the models' ability to create a NLDB to separate the two classes in the presence of noise. The data is generated randomly in the shape of two bananas facing each other, corresponding to the two data classes. Each class contains 90 data points. The experiment is conducted in three scenarios: the original data without any noise, data with 10% noise added, and data with 25% noise.

The experimental setup for the Double Banana is similar to the previous experiment (sub-section 5.2.1). Because the decision boundary in the Double Banana data set is non-linear, the model is trained to obtain two optimum parameters (λ and γ). The model is trained and validated over 20 iterations, and the accuracy is averaged over those 20 iterations, as explained earlier. The optimum λ^* and γ^* are then obtained, and the model is solved with the optimum λ^* and γ^* using the training and validations sets. As a result, the optimum solution $\boldsymbol{u}^*$ and b^*(or $\boldsymbol{u}^*$ only for C-PSVM model) are then obtained.

It was discussed in sub-section 3.2.2 that the NLDB is not represented by a non-linear function. This may raise questions about the mechanism of plotting the NLDB on a 2D screen. This is simply done by equating the decision boundary function to zero and solving for (x_1, x_2). Note that, there are an infinite number of solutions to this equation. Therefore, the easiest way to practically plot the NLDB is to create a 2D grid of points (mesh) with a relatively high resolution, and substitute each point from the grid in the decision boundary function. If the result equals zero or very close to zero, then it can be safely concluded that the point belongs to the decision boundary. Those points belonging to the decision boundary are then plotted to construct the NLDB.

Figures (34)-(38) show the data sets and decision boundary for SVM, PSVM, C-SVM, and C-PSVM, respectively. Each figure is divided into 3 rows and 2 columns. The columns represent the 3 different scenarios (i.e. no noise added, 10% noise, and 25% noise, respectively). The first column shows the test sample only against the final decision boundary. Note the test sample determines the prediction accuracy and therefore, a perfect separation of the test sample indicates a 100% prediction accuracy. The second column shows the decision boundary with the training and validation data set.

When the decision boundary is continuous as shown in Figure (39), the domain is divided into two distinct areas (A and B), and each area represents the domain of one class. When the data points of the two classes are linearly or non-linearly separable, a continuous decision boundary can always be obtained. However, when the data sets are not separable, then the decision boundary is always exposed to experience a discontinuous behavior to accommodate the points which migrate into the other class's vicinity. This is also known as overfitting. The higher a model's sensitivity to outliers the higher the probability is to experience overfitting, thus a discontinuous decision boundary. Figure (40) shows an example of a discontinuous decision boundary, where each class domain is represented by multiple areas.

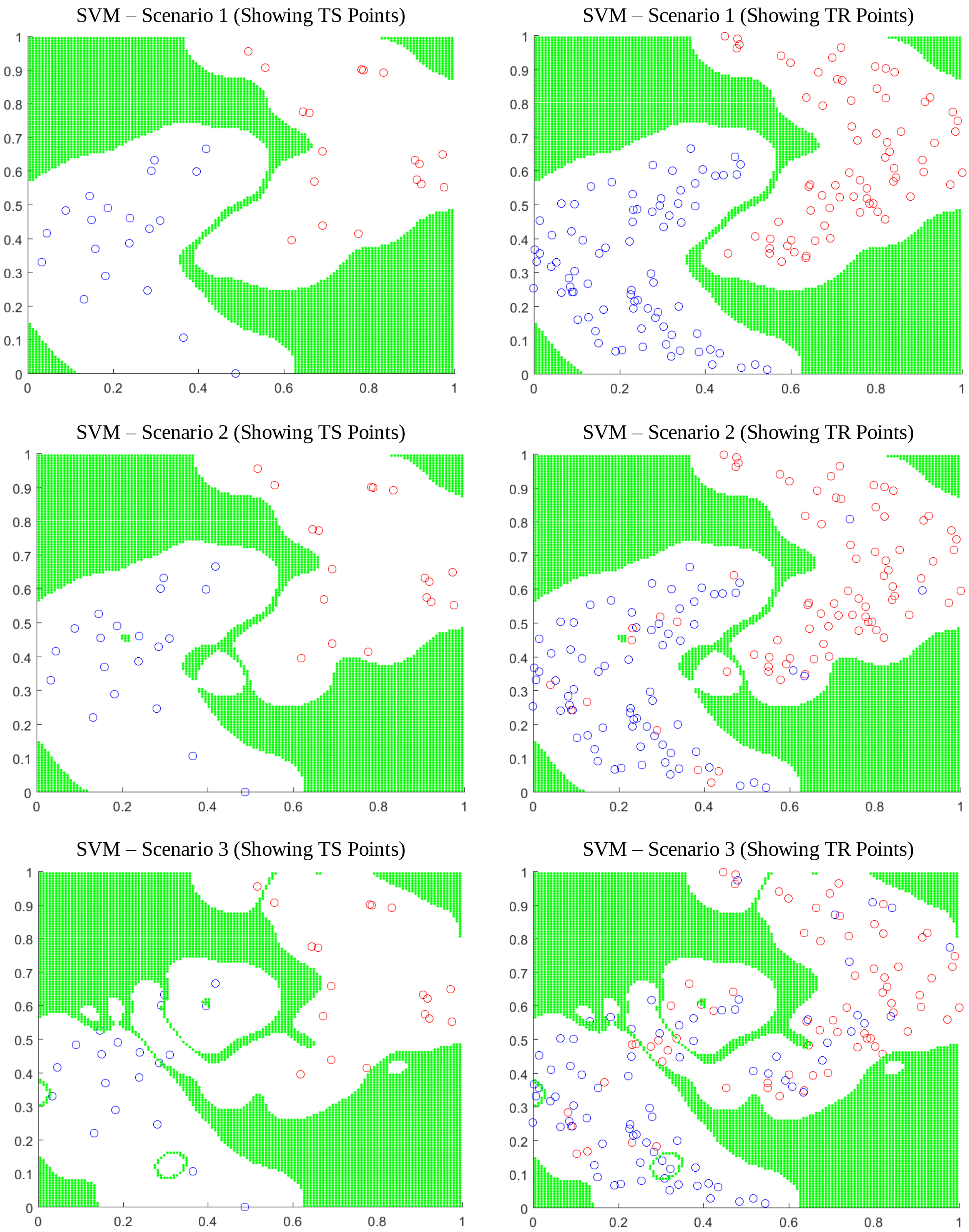


Figure 34: The NLDB obtained by the SVM for the 3 scenarios

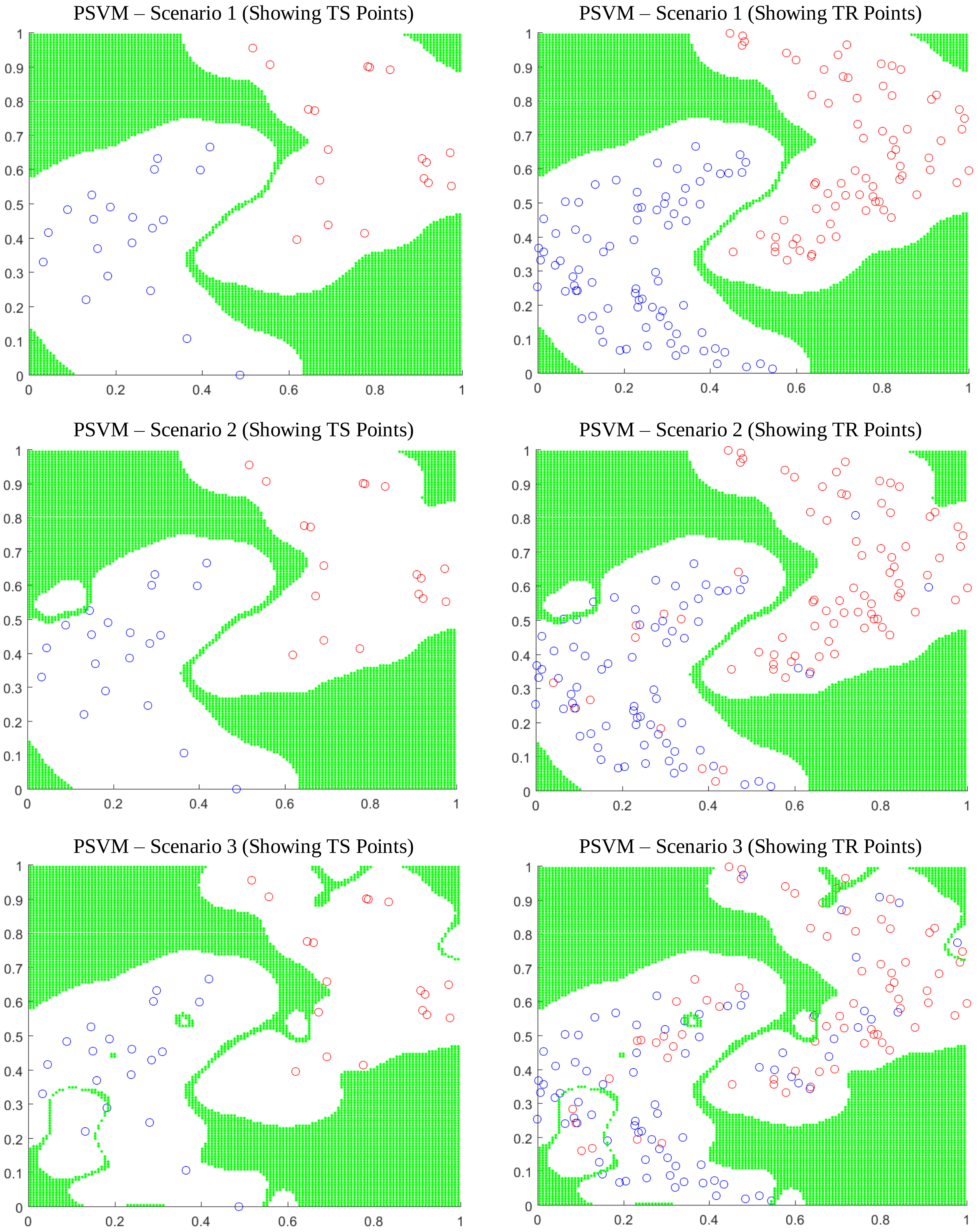
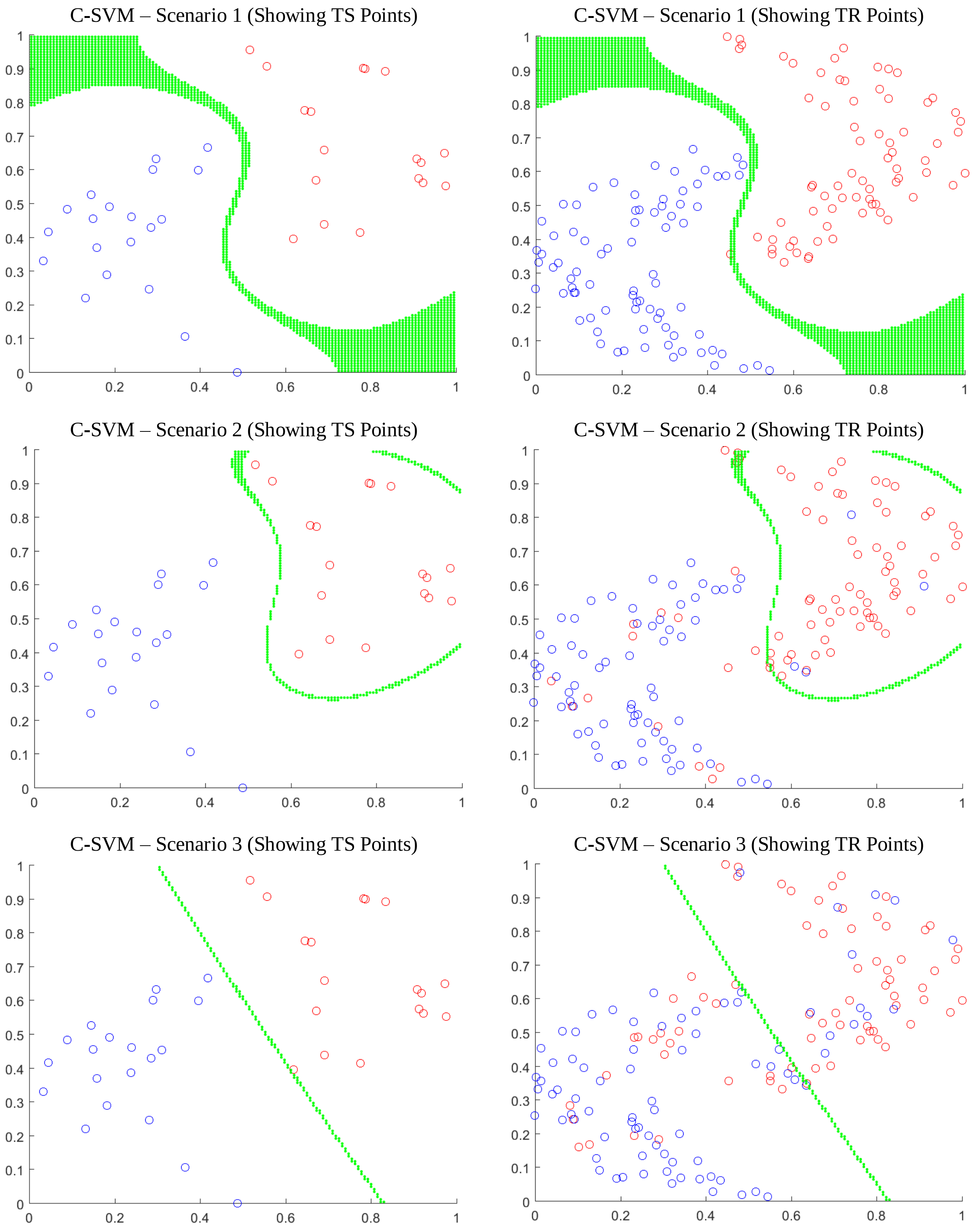


Figure 35: The NLDB obtained by the PSVM for the 3 scenarios


C-SVM – Scenario 1 (Showing TS Points)

C-SVM – Scenario 1 (Showing TR Points)

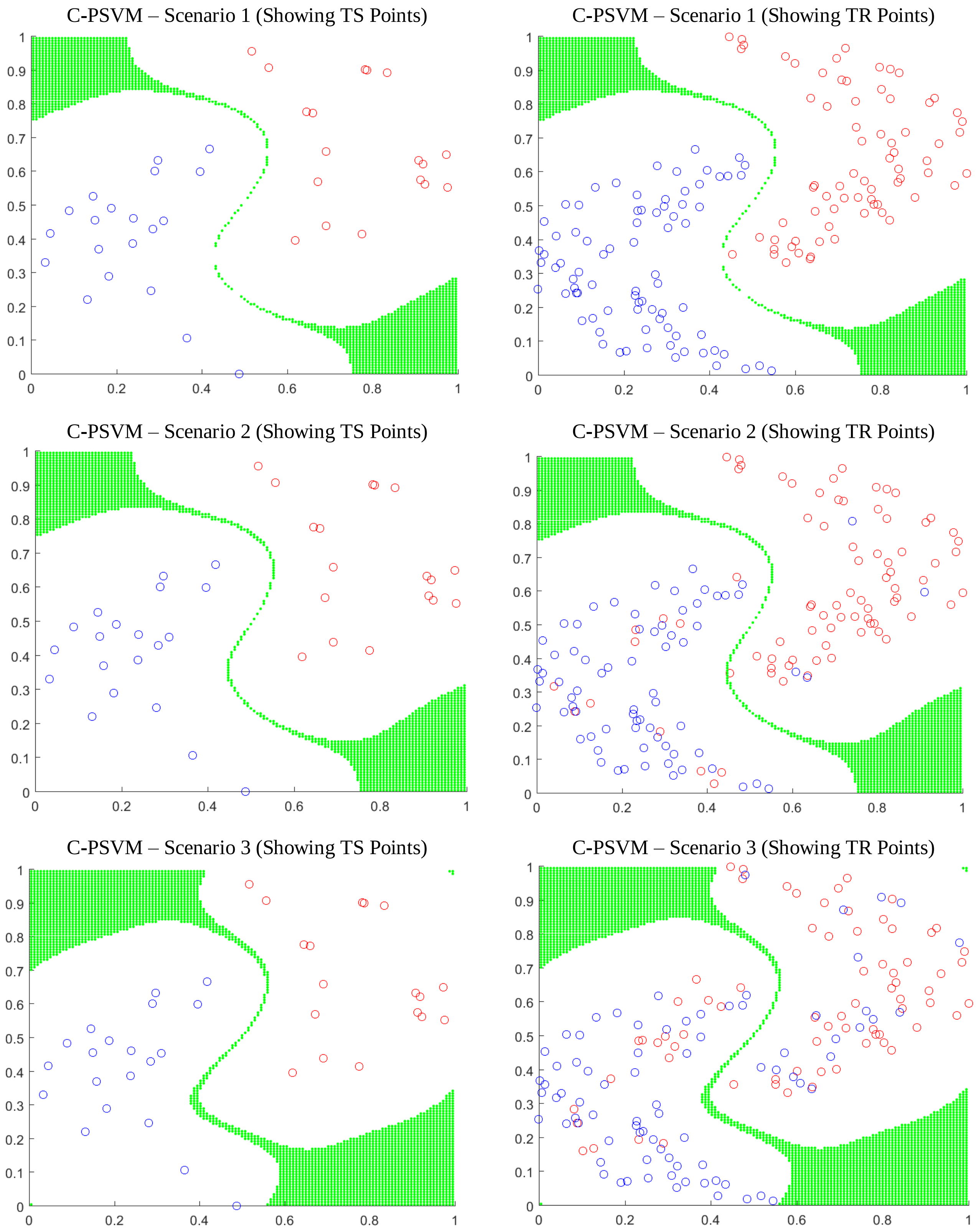
C-SVM – Scenario 2 (Showing TS Points)

C-SVM – Scenario 2 (Showing TR Points)

C-SVM – Scenario 3 (Showing TS Points)

C-SVM – Scenario 3 (Showing TR Points)

Figure 36: The NLDB obtained by the C-SVM for the 3 scenarios



C-PSVM – Scenario 1 (Showing TS Points)

C-PSVM – Scenario 1 (Showing TR Points)

C-PSVM – Scenario 2 (Showing TS Points)

C-PSVM – Scenario 2 (Showing TR Points)

C-PSVM – Scenario 3 (Showing TS Points)

C-PSVM – Scenario 3 (Showing TR Points)

Figure 37: The NLDB obtained by the C-PSVM for the 3 scenarios

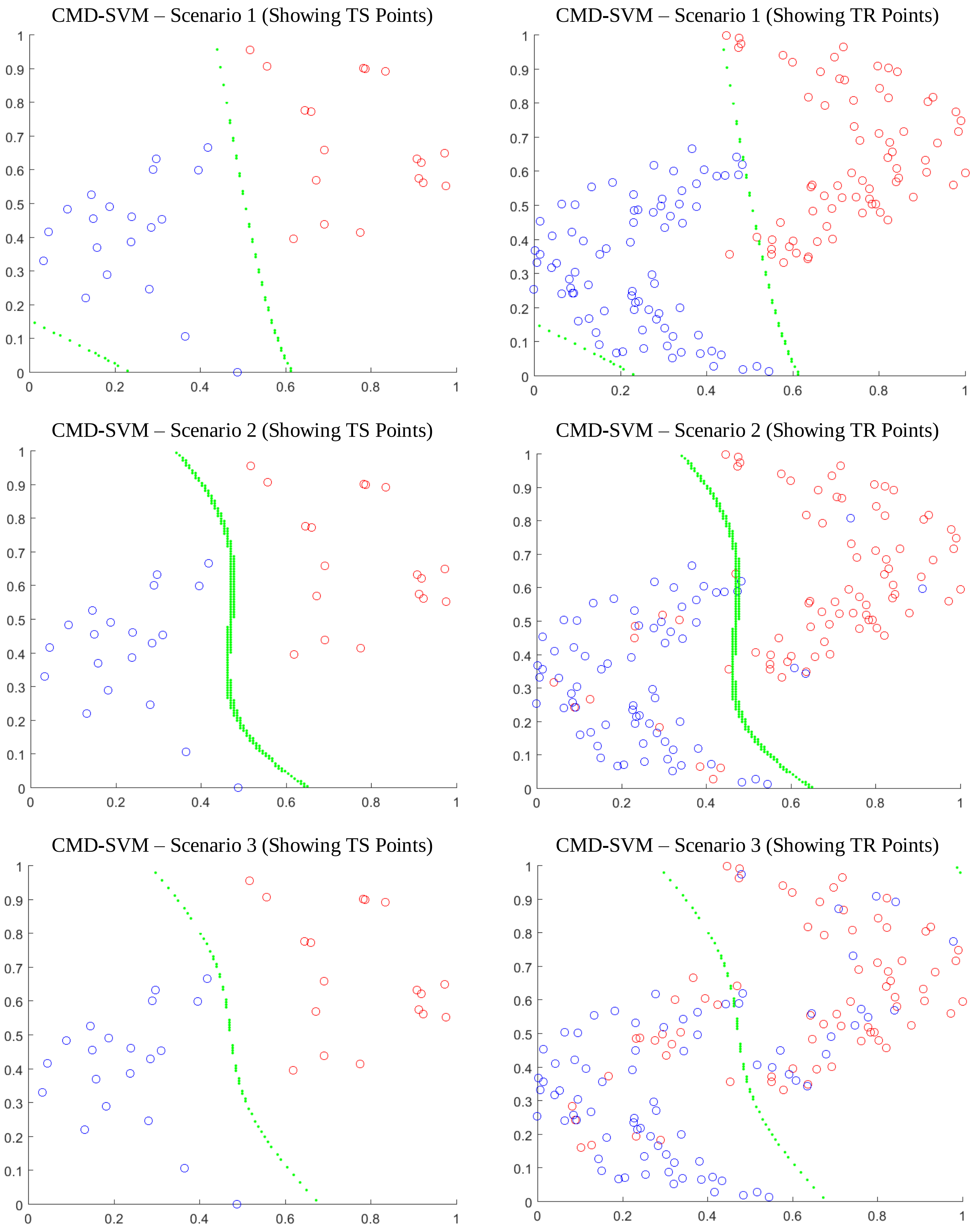


Figure 38: The NLDB obtained by the CMD-SVM for the 3 scenarios

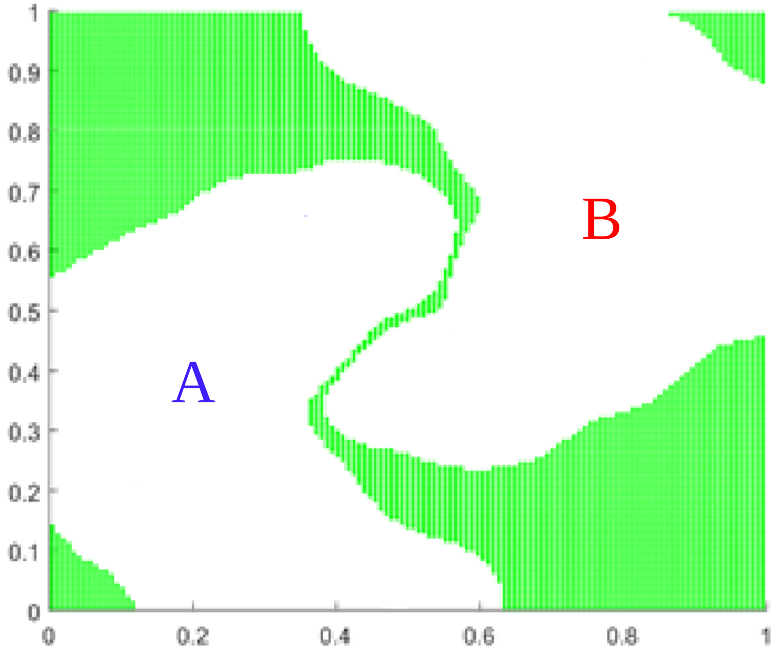


Figure 39: An example of a continuous NLDB

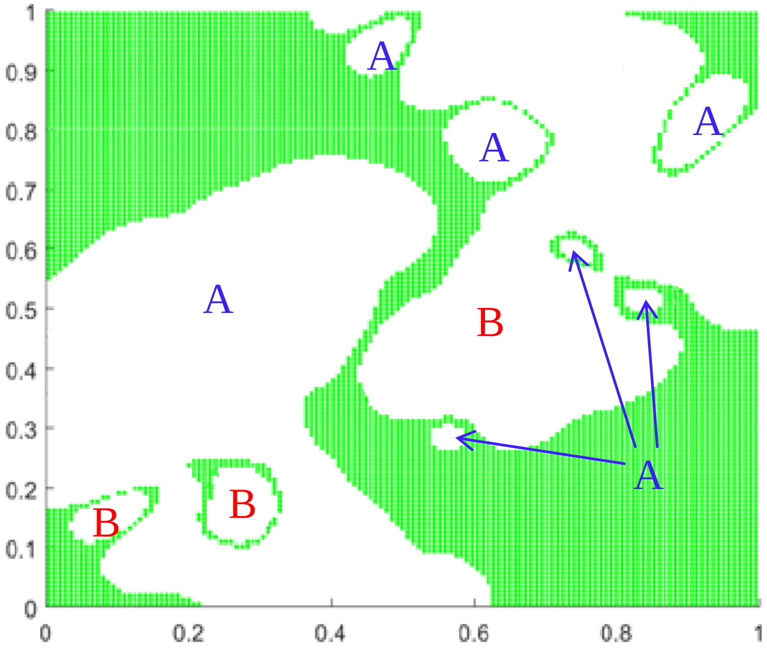


Figure 40: An example of a discontinuous NLDB and the relevant classes

All 4 models delivered similar results in Scenario 1. In Scenario 2 and 3, the proposed robust models outperformed the SVM and PSVM models in noise insensitivity (robustness). From Figures (34)-(38), it is observed that the NLDBs obtained by the conventional SVM models experience higher discontinuity as the noise level increases. The discontinuity of a decision boundary can be related to overfitting. Generally, when a model overfits, the resultant NLDB becomes discontinuous, similar to the one shown in Figure (40). This observation of overfitting can be related to the model's sensitivity to noise. When the model is noise sensitive, it will try to accommodate every violating point into its associated class. Consequently, NLDBs will be created to isolate those violating points from the other class's points near them. In this case, the validation accuracy tends to be high, whereas the prediction accuracy using the testing data is expected to be significantly lower.

5.4 Real-Life Problems

The main purpose of Section 5.3 is to illustrate the behavior of the decision boundaries obtained by various models, when noise is added to the clean (noise free) synthetic data. However, those data sets may not represent real-life data scenarios. Normally, real-life data sets contain some natural noise, which is extremely difficult to identify straightforward. Typically, real-life data sets come in higher feature space dimension ($m \geq 3$), which influences the model behavior due to the curse of dimensionality [1].

These reasons highlight the importance of solving real-life problems to judge the models for robustness.

The data sets used in this section are downloaded from the UCI data repository [33]. A total of 8 well-known data sets were selected to run the experiment, which are described in Table (3). It was observed that solving the CMD-SVM model takes a tremendous amount of time due to the additional nested loop for α. Therefore, the CMD-SVM model is excluded from the experiments, and the remaining four models are evaluated in this section.

Table 3: Details of the real-life data sets

No.	Data Set Name	Number of points (n)	Number of features (m)	Missing Values?	Number of removed data instances
1	SPECTF Heart	267	44	No	N/A
2	Breast	699	10	Yes	16
3	Sonar	208	60	No	N/A
4	Haber	306	3	No	N/A
5	Inosphere	351	34	No	N/A
6	Monks1	432	6	No	N/A
7	Monks2	432	6	No	N/A
8	Monks3	432	6	No	N/A

All the experiments in this section are carried out in 3 scenarios. In the first scenario, the experiment is run with the original data without any noise added. The second and third scenarios will have the same original data set but with 10% and 20% noise added, respectively. In each scenario, the complete CV process is repeated 20 times and the prediction accuracy measure is averaged across the 20 repetitions. The prediction accuracies resulted from the experiments using the real-life data sets are shown in Table

(4). Moreover, the highest accuracies are shown in bold. As seen from the table, the highest accuracies are mainly obtained by the C-SVM and C-PSVM.

Figure (41) provides a different insight on each model's performance in addition to a visual comparison among the 4 models. One interpretation of Figure (41) is that the smaller the change over the noise level (x-axis) the higher is the robustness. Additionally, it is easier to compare the 4 models in terms of their prediction accuracies, as the higher the line the higher is the accuracy. Generally, it can be seen that the proposed robust models experience a smaller change across the noise levels, whereas the SVM and PSVM appear to have steeper negative slopes.

The proposed robust models demonstrated higher prediction accuracy compared to the conventional SVM models. Generally, the C-SVM model outperformed all the models in terms of prediction accuracy. The C-SVM scored the highest prediction accuracy 62.5% of the time and the C-PSVM scored the highest 37.5% of the time.

Table 4: The prediction accuracies obtained by the 4 models for the 3 scenarios, using the real-life data sets.

No Noise Added

Dataset	SVM	PSVM	r_hinge-SVM [16]	C-SVM	C-PSVM
SPECTF	0.7905 ± 0.059	0.8160 ± 0.062	0.7849 ± 0.078	**0.8458 ± 0.046**	0.8383 ± 0.047
Breast	0.9430 ± 0.019	0.9583 ± 0.015	0.9510 ± 0.014	0.9487 ± 0.02	**0.9750 ± 0.015**
Sonar	0.7711 ± 0.083	0.8679 ± 0.047	0.7674 ± 0.078	0.8606 ± 0.032	**0.8791 ± 0.047**
Haber	0.7350 ± 0.054	0.7281 ± 0.047	0.7282 ± 0.066	**0.7806 ± 0.032**	0.7354 ± 0.052
Ionosphere	0.7142 ± 0.033	0.8747 ± 0.037	0.7172 ± 0.045	0.8632 ± 0.03	**0.8854 ± 0.031**
Monks1	0.7604 ± 0.065	0.7750 ± 0.042	0.7771 ± 0.063	**0.8170 ± 0.044**	0.8164 ± 0.025
Monks2	0.7771 ± 0.046	0.7767 ± 0.050	0.7644 ± 0.047	**0.8211 ± 0.035**	0.8089 ± 0.026
Monks3	0.8756 ± 0.040	0.8852 ± 0.027	0.9004 ± 0.028	**0.9219 ± 0.029**	0.8882 ± 0.036

10% Noise Added

Dataset	SVM	PSVM	r_hinge-SVM [16]	C-SVM	C-PSVM
SPECTF	0.7624 ± 0.073	0.8308 ± 0.051	0.7675 ± 0.083	**0.8535 ± 0.047**	0.8336 ± 0.048
Breast	0.9382 ± 0.026	0.9502 ± 0.019	0.9557 ± 0.021	0.9528 ± 0.016	**0.9762 ± 0.016**
Sonar	0.7128 ± 0.127	0.8019 ± 0.074	0.7550 ± 0.070	0.8400 ± 0.052	**0.8403 ± 0.048**
Haber	0.7424 ± 0.04	0.7264 ± 0.047	0.7596 ± 0.064	**0.7618± 0.050**	0.7476 ± 0.052
Ionosphere	0.7085 ± 0.054	0.8314 ± 0.048	0.7270 ± 0.043	0.8383 ± 0.043	**0.8662 ± 0.030**
Monks1	0.7020 ± 0.062	0.7268 ± 0.046	0.7297 ± 0.044	0.7881 ± 0.036	**0.7933 ± 0.049**
Monks2	0.7251 ± 0.063	0.7222 ± 0.062	0.7273 ± 0.045	**0.8002 ± 0.032**	0.7545 ± 0.056
Monks3	0.8120 ± 0.071	0.8602 ± 0.032	0.8487 ± 0.032	**0.8831 ± 0.034**	0.8692 ± 0.052

20% Noise Added

Dataset	SVM	PSVM	r_hinge-SVM [16]	C-SVM	C-PSVM
SPECTF	0.7180 ± 0.130	0.7805 ± 0.072	0.7678 ± 0.123	**0.8451 ± 0.041**	0.8243 ± 0.051
Breast	0.9367 ± 0.021	0.9451 ± 0.027	0.9443 ± 0.020	0.9502 ± 0.018	**0.9788 ± 0.012**
Sonar	0.6715 ± 0.095	0.7476 ± 0.065	0.7170 ± 0.084	**0.8312 ± 0.060**	0.8203 ± 0.058
Haber	0.7180 ± 0.069	0.7297 ± 0.049	0.7270 ± 0.062	**0.7642 ± 0.043**	0.7435 ± 0.046
Ionosphere	0.6956 ± 0.055	0.8147 ± 0.045	0.7165 ± 0.046	0.8341 ± 0.044	**0.8527 ± 0.041**
Monks1	0.6568 ± 0.053	0.6542 ± 0.076	0.7132 ± 0.042	**0.7557 ± 0.030**	0.7372 ± 0.068
Monks2	0.6749 ± 0.055	0.6828 ± 0.048	0.7057 ± 0.52	**0.7487 ± 0.045**	0.7441 ± 0.048
Monks3	0.7877 ± 0.079	0.8068 ± 0.061	0.8082 ± 0.038	**0.8721 ± 0.043**	0.8553 ± 0.036

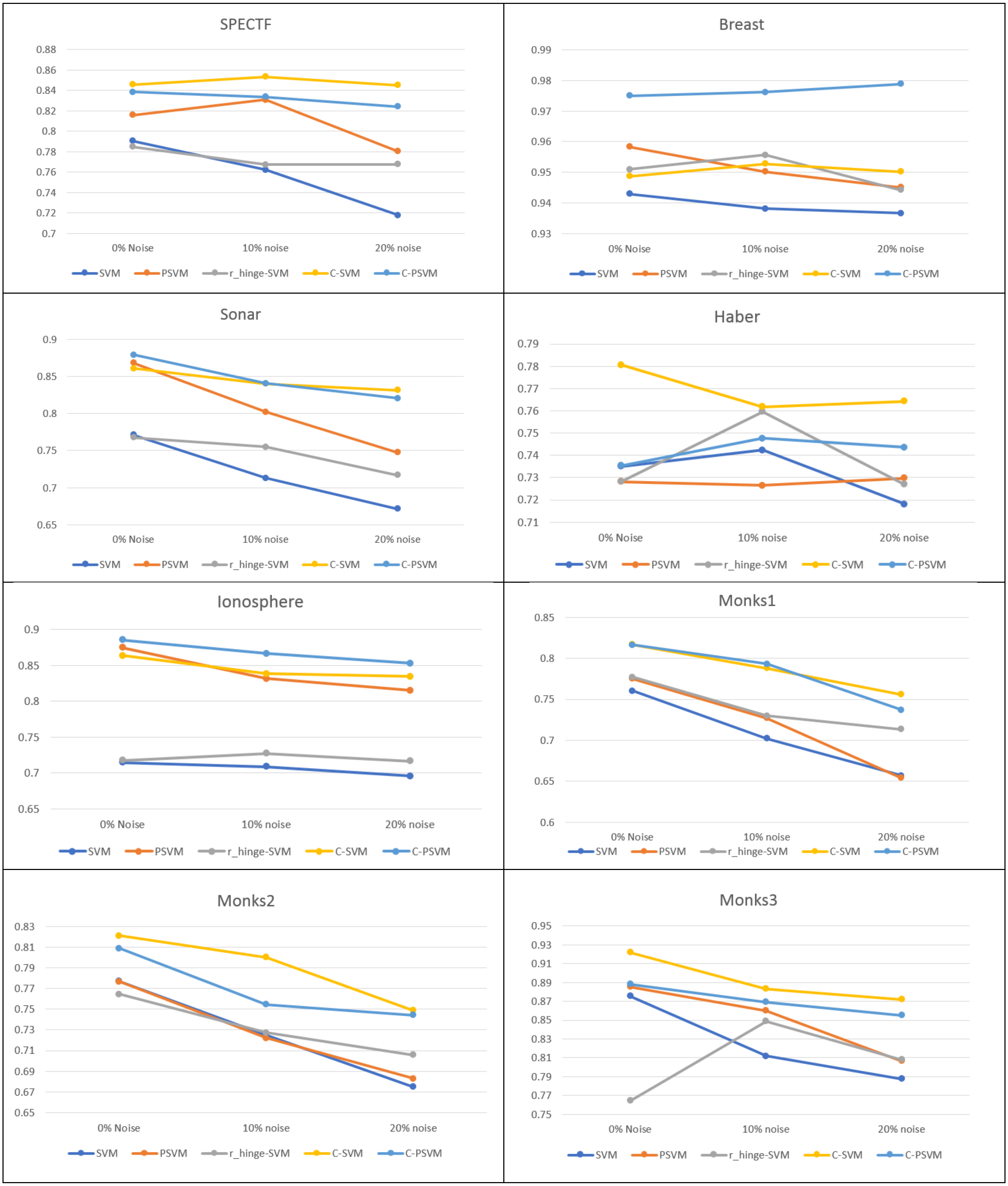


Figure 41: Line chart of the prection accuracies across the noise levels and per data set

Although the focus of this thesis is the accuracy as a performance measure, the solution time remains a critical issue. From Chapter (4), the proposed solution methodology introduces the parameter σ, which imposes an additional iteration on the solution algorithm. Therefore, it is obvious to notice that the proposed models would take more time to solve compared to the SVM and PSVM. Table (5) illustrate the solution time in seconds for one complete repetition of each experiment. For $\sigma_s = 30$, $\sigma_e = 0.1$, and $\tau = 0.9$, a total of 55 values for σ are to be evaluated. From the last two columns of the same table, it can be observed that the solution time for a robust model is close to 55 times the solution time of its associated conventional (noise sensitive) model. This indicates that the solution time of a robust model for a specific value of σ is comparable to the solution time of the associated conventional model for one repetition.

Based on the discussion so far and the results shown in Table (5), the relatively longer solution time for the robust models are caused by the handling of σ. This can be verified by observing the time for solving only the NLP model. Regarding the proposed robust models, the solution time for every value of σ was recorded, and the figures shown in Table (6) are the average values in seconds. It can be seen that the average time it takes to solve the NLP problem of the proposed robust models is comparable or better than the conventional models. The reason for that is the iterative solution methodology. In every iteration, the initial solution obtained from the previous iteration is always in the valley, hence closer to the optimum solution. Therefore, the average solution time for one σ is less compared to the models that are not solved by the proposed iterative methodology. However, the total time to solve the proposed models using the iterative solution methodology is higher due to the ramification of σ.

Table 5: The total solution time in seconds for one complete repetition

0% Noise

Dataset	SVM	PSVM	C-SVM	C-PSVM	Ratio (C-SVM:SVM)	Ratio (C-PSVM:PSVM)
SPECT	9.47	9.94	463.22	459.79	49	46
Breast	120.73	112.84	4091.64	4048.74	34	36
Sonar	18.80	36.67	689.05	675.04	37	18
Haber	11.72	11.12	615.89	556.56	53	50
Inosphere	13.55	15.38	881.81	735.96	65	48
Monks1	19.13	24.90	1301.00	939.72	68	38
Monks2	19.08	22.90	1289.48	952.89	68	42
Monks3	20.59	21.88	1424.72	960.70	69	44
				Average	55	40

10% Noise

Dataset	SVM	PSVM	C-SVM	C-PSVM	Ratio (C-SVM:SVM)	Ratio (C-PSVM:PSVM)
SPECT	19.43	19.15	1010.91	924.53	52	48
Breast	238.58	226.55	9663.79	1347.62	41	6
Sonar	36.95	54.27	1394.14	9559.39	38	176
Haber	22.70	22.65	1239.18	1103.48	55	49
Inosphere	28.32	29.60	1744.55	1480.61	62	50
Monks1	38.55	46.14	2689.07	1880.82	70	41
Monks2	38.62	43.73	2551.02	1907.27	66	44
Monks3	43.46	42.70	2799.61	1954.66	64	46
				Average	56	57

20% Noise

Dataset	SVM	PSVM	C-SVM	C-PSVM	Ratio (C-SVM:SVM)	Ratio (C-PSVM:PSVM)
SPECT	28.75	29.44	1556.43	1386.85	54	47
Breast	356.95	339.09	15240.15	2020.92	43	6
Sonar	53.99	71.84	2073.89	15073.51	38	210
Haber	33.92	36.55	1805.79	1641.14	53	45
Inosphere	42.50	44.12	2566.01	2162.56	60	49
Monks1	57.96	67.00	4097.33	2818.23	71	42
Monks2	58.91	67.75	3818.64	2861.71	65	42
Monks3	65.77	62.50	4189.74	2917.50	64	47
				Average	56	61

Table 6: The time (average time for C-SVM and C-PSVM) taken to solve the NLP problem for once

0% Noise

Dataset	SVM	PSVM	r_hinge-SVM [16]	C-SVM	C-PSVM
SPECT	**2.49**	2.55	2.98	3.45	2.73
Breast	54.35	52.29	45.30	**5.61**	10.41
Sonar	**4.53**	4.74	5.52	5.78	5.30
Haber	3.27	3.37	3.22	4.11	**2.76**
Inosphere	4.33	4.45	6.52	5.40	**3.03**
Monks1	6.83	8.59	8.21	9.07	**3.29**
Monks2	6.82	7.23	9.52	9.04	**3.21**
Monks3	6.98	7.31	9.77	10.20	**3.20**

10% Noise

Dataset	SVM	PSVM	r_hinge-SVM [16]	C-SVM	C-PSVM
SPECT	3.21	**2.56**	3.36	3.49	3.11
Breast	53.13	52.91	44.02	**5.67**	10.50
Sonar	**4.47**	4.62	5.26	5.52	5.36
Haber	3.84	3.93	3.88	4.37	**2.74**
Inosphere	4.35	4.44	6.61	5.44	**3.06**
Monks1	6.85	6.97	8.49	9.87	**3.29**
Monks2	6.80	7.11	9.22	9.24	**3.20**
Monks3	7.12	7.19	9.03	10.49	**3.17**

20% Noise

Dataset	SVM	PSVM	r_hinge-SVM [16]	C-SVM	C-PSVM
SPECT	**2.49**	2.54	2.57	2.92	2.77
Breast	52.46	51.98	42.98	**5.52**	10.28
Sonar	**4.51**	4.66	5.40	5.33	5.21
Haber	4.56	5.45	5.09	3.25	**2.77**
Inosphere	4.41	4.68	6.41	5.64	**2.81**
Monks1	6.85	6.84	9.36	10.72	**3.24**
Monks2	7.31	7.14	10.22	8.96	**3.18**
Monks3	9.34	7.05	9.54	11.06	**3.13**

The C-SVM and C-PSVM scored the lowest solution time for a given σ 75% of the time. Furthermore, the main advantage of the PSVM model (solution speed) is retained after the incorporation of the CLF. As a result, the C-PSVM generally outperformed all the other models. This shows that the incorporation of the CLF into the proposed models does not have a negative impact on the solution time.

It is important to note that the NLP problems are solved using the Grid & Multi-Threading facility, which means the figures shown on the following tables are for the longest thread. Additionally, these figures include the communication time from Matlab to GAMS, from GAMS to SNOPT solver, and vice versa. Furthermore, all the computations were carried out on the same equipment.

CHAPTER 6

DISCUSSION AND CONCLUSION

In this chapter, a discussion on the proposed models and their applicability in data classification is presented, followed by thesis conclusion. Lastly, some opportunities for future research are also proposed.

6.1 Discussion and Highlights

6.1.1 Robustness in Classification

Robustness in data analytics is defined as the insensitivity of the method w.r.t noise in the given data set. Noise may enter into any real data set due to several reasons. For example: noise may exist due to improper methods of gathering the data, inaccurate class labeling of gathered data, errors during data handling, etc. Noise in data is normally uneasy to identify by a simple procedure.

Robust classification models have become high in demand in today’s sophisticated applications [21][22][23]. The presence of noise in the data can significantly affect the obtained classifier, thus lead to inaccurate classification of new instances. This can be critical in some areas of applications such as those related to high cost impact, high risk, or human safety. However, robust classification models are typically computationally expensive when compared to their non-robust counterparts. In this thesis, Correntropic Loss Function (CLF) [24] based robust classification models is proposed.

6.1.2 The Special Effect of σ

The CLF is used in the proposed models to induce robustness. The penalty that is assigned by the CLF starts from zero and then nonlinearly increases as the misclassification error increases until a certain error value, where it sluggishly increasing (practically remains almost flat). The most critical parameter in CLF is σ, which is a tuning parameter that controls the shape of the function. For small values of σ, the function tends to be locally pseudo-convex, but for large values it tends to be convex. This characteristic is exploited in the proposed solution methodology by introducing the proposed iterative solution method. This proposed method ensures the obtained solution across all iterations is always retained inside the valley and guarantee a local minimum. The slower the rate of change in σ the smoother is the change in the decision boundary and less likely to escape the valley domain.

6.1.3 The Trade-off Between Time and Robustness

For any classification problem, each performance measure has different importance or relevance depending on the application and the user’s objective. For example, some users require higher solution speed, but others demand higher accuracy. Similarly, some classification models excel in delivering more accurate results, but others outperform in solution time. An example of this was discussed in Chapter 2 when the SVM and PSVM models were compared [7].

As shown in Chapter 5, the proposed robust models deliver more accurate results under the existence of outliers. However, there is a cost for that improved performance, and it is related to the solution time. The addition of the correntropic parameter σ created an additional line search, which contributed to the total solution time. As explained in Section 4.3, the finer the search set for σ, the higher the probability to have the iterative solution in the valley domain. Furthermore, the decision boundary will smoothly change over the search set. Therefore, this behavior obtained by a finer search set is highly preferred. However, it is a tradeoff w.r.t the solution speed.

6.2 Conclusion

In this thesis, a total of 3 robust SVM models for binary classification were developed using the CLF. The CLF has unique characteristics, which differentiate the associated robust models from the others. The most important characteristic is the special effect of σ, which was exploited to develop an iterative solution methodology. In Section 4.3, it was experimentally and visually illustrated that the proposed solution methodology offers a superior performance in solving the proposed robust models when compared to non-iterative methodology.

From Chapter 5, it can be concluded that the models are robust and less sensitive to the outliers. Moreover, the proposed models generally produce better accuracies regardless of the noise level. However, this admired performance compromises the solution time. Since the proposed models contain an additional parameter (σ), the additional time needed to evaluate the different values of σ is almost inevitable. The trade-off between accuracy and solution time needs to be managed by appropriately selecting the range for σ (σ_s and σ_e) and the reduction parameter τ.

6.3 Future Research

In this section, some ideas are proposed for future research which may potentially extend or improve the work in this thesis.

6.3.1 GRT versus The Dual Formulation

From the literature, the kernelized SVM models are typically developed from the dual formulation, as explained in Sub-sections 3.2.1 and 3.2.2. The kernelized robust models proposed in this thesis are developed using the Generalized Representer Theorem (GRT). The latter does not require the dual formulation as it derives the kernelized model from the primal formulation. The kernelized dual formulation for the proposed robust models should have a different structure. The new model structure may possess different characteristics, which would possibly yield to smaller problem dimension, and/or higher solution speed.

6.3.2 A Closed Form for Determining $\boldsymbol{\sigma_s}, \boldsymbol{\sigma_e}$, and $\boldsymbol{\tau}$

In Section 4.3, it is explained that determining σ_s, σ_e and τ is critical to ensure the problem is solved efficiently and a local optimum is always obtained. The trial and error method is typically inefficient, and if the user is solving different models for multiple data sets, then assigning those values efficiently may become critical.

To swiftly determine σ_s and σ_e, it is always safe to set σ_s to a very large value and σ_e to be very low. The drawback of starting from a massively large σ and ending at an extremely small value is the number of iterations the algorithm would take to evaluate all these values of σ. The interaction between the 3 values creates a trade-off between solution optimality and solution time.

It was found by [27] that σ_s and σ_e in the CLF can be safely predetermined by simply measure the maximum and minimum distance between any two points, respectively. This rule cannot be simply applied in the proposed robust models because the loss function was slightly modified to incorporate into the model structure. Moreover, the objective function in the proposed models contains the regularization term $\|\mathrm{w}\|^2$. Therefore, a similar approach to predetermine σ_s, σ_e and τ is expected to be very valuable.

References

[1] C. M. Bishop, *Pattern Recognition and Machine Learning*, vol. 53, no. 9. 2013.

[2] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From Data Mining to Knowledge Discovery in Databases," *AI Mag.*, vol. 17, no. 3, p. 37, 1996.

[3] L. Hamel, *Knowledge Discovery with Support Vector Machines*. 2009.

[4] M. K. Keleş, "AN OVERVIEW : THE IMPACT OF DATA MINING APPLICATIONS ON VARIOUS SECTORS," vol. 6168, pp. 128–132, 2017.

[5] S. B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques," *Informatica*, vol. 31, pp. 249–268, 2007.

[6] C. Cortes and V. Vapnik, "Support-Vector Networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[7] G. Fung, O. L. Mangasarian, and W. D. Street, "Proximal Support Vector Machine Classifiers," pp. 77–86.

[8] J. A. K. Suykens and J. Vandewalle, "Least Squares Support Vector Machine Classifiers," *Neural Process. Lett.*, vol. 9, no. 3, pp. 293–300, 1999.

[9] C. Lin and S. Wang, "Fuzzy support vector machines," *Neural Networks, IEEE Trans.*, vol. 13, no. 2, pp. 464–471, 2002.

[10] Z. Meng, F. Li-hua, W. Gao-feng, and H. Ji-cheng, "Weighted Proximal Support Vector Machines: Robust Classification," *Wuhan Univ. J. Nat. Sci.*, vol. 10, no. 3, pp. 507–510, 2005.

[11] G. Heo and P. Gader, "Fuzzy SVM for noisy data: A robust membership calculation method," in *IEEE International Conference on Fuzzy Systems*, 2009, no. 1, pp. 431–436.

[12] Y. Wang, S. Wang, and K. K. Lai, "A new fuzzy support vector machine to evaluate credit risk," *IEEE Trans. Fuzzy Syst.*, vol. 13, no. 6, pp. 820–831, 2005.

[13] X. Yang, L. Han, Y. Li, and L. He, "A bilateral-truncated-loss based robust support vector machine for classification problems," *Soft Comput.*, vol. 19, no. 10, pp. 2871–2882, 2015.

[14] Y. Wu and Y. Liu, "Robust truncated hinge loss support vector machines," *J. Am. Stat. Assoc.*, vol. 102, no. 479, pp. 974–983, 2007.

[15] Y. Feng, Y. Yang, X. Huang, S. Mehrkanoon, and Johan A. K. Suykens, "Robust Support Vector Machines for Classification with Nonconvex and Smooth Losses," *Neural Comput.*, vol. 28, pp. 1217–1247, 2016.

[16] G. Xu, Z. Cao, B. G. Hu, and J. C. Principe, “Robust support vector machines based on the rescaled hinge loss function,” *Pattern Recognit.*, vol. 63, no. March 2016, pp. 139–148, 2017.

[17] Y. Ma, L. Li, X. Huang, and S. Wang, “Robust Support Vector Machine using Least Median Loss Penalty,” in *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 2011, vol. 18, no. PART 1, pp. 11208–11213.

[18] L. Wang, H. Jia, and J. Li, “Training robust support vector machine with smooth Ramp loss in the primal space,” *Neurocomputing*, vol. 71, no. 13–15, pp. 3020–3025, 2008.

[19] P. K. Shivaswamy and T. Jebara, “Relative margin machines,” *Adv. Neural Inf. Process. Syst.*, vol. 21, pp. 1–8, 2008.

[20] Y. Song, W. Zhu, Y. Xiao, and P. Zhong, “Robust relative margin support vector machines,” *J. Algorithms Comput. Technol.*, vol. 11, no. 2, pp. 186–191, 2017.

[21] A. Takeda, S. Fujiwara, and T. Kanamori, “Extended Robust Support VectorMachine Based on Financial Risk Minimization,” *Neural Comput.*, vol. 26, pp. 2541–2569, 2014.

[22] X. Lu, W. Liu, C. Zhou, and M. Huang, “Probabilistic weighted support vector machine for robust modeling with application to hydraulic actuator,” *IEEE Trans. Ind. Informatics*, vol. 13, no. 4, pp. 1723–1733, 2017.

[23] Z. Gu, Z. Zhang, J. Sun, and B. Li, “Robust image recognition by L1-norm twin-projection support vector machine,” *Neurocomputing*, vol. 223, no. October 2015, pp. 1–11, 2017.

[24] A. Singh and J. C. Principe, “A loss function for classification based on a robust similarity metric,” in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, 2010, no. X, pp. 1–6.

[25] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[26] B. Schölkopf, R. Herbrich, and A. J. Smola, “A Generalized Representer Theorem,” pp. 416–426, 2001.

[27] M. N. Syed, P. M. Pardalos, and J. C. Principe, “On the optimization properties of the correntropic loss function in data analysis,” *Optim. Lett.*, vol. 8, no. 3, pp. 823–839, 2014.

[28] O. L. Mangasarian, “PSEUDO-CONVEX FUNCTIONS,” *J.SIAM Control*, vol. 3, no. 2, pp. 281–290, 1965.

[29] Z. Slodkowski, “Pseudoconvex classes of functions. I. Pseudoconcave and pseudoconvex sets,” *Pacific J. Math.*, vol. 134, no. 2, pp. 343–376, 1988.

[30] P. Škrabánek and P. Doležel, “On Reporting Performance of Binary Classifiers,” *Sci. Pap. Univ. Pardubice. Ser. D, Fac. Econ. Adm.*, vol. 25, no. 41, pp. 181–192, 2017.

[31] M. N. Syed, J. C. Principe, and P. M. Pardalos, *Correntropy in Data Classification Mujahid*, vol. 20. Dynamics of Information Systems: Mathematical Foundations, 2012.

[32] X. H. Cao, I. Stojkovic, and Z. Obradovic, “Open Access A robust data scaling algorithm to improve classification accuracies in biomedical data,” pp. 1–11, 2016.

[33] D. Dua and E. Karra Taniskidou, “UCI Machine Learning Repository,” *Irvine, CA: University of California, School of Information and Computer Science*, 2017. [Online]. Available: http://archive.ics.uci.edu/ml.

Vitae

Name :Mohammed M. Al-Mehdhar

Nationality :Yemeni

Date of Birth :12/1/1987

Email :almehdar@hotmail.com

Address :Dammam, Kingdom of Saudi Arabia

Academic Background :Industrial and Systems Engineering

Job Experience :Process Optimization, Project Management, and Strategic Planning

Research Interest :Data Analytics, and Deterministic Supply Chain Models